Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

Concurrency specification using Event-based Specification Chart

Dumitru Ciorbă, Victor Beşliu

Abstract

Architecting framework proposed in [1] can be used efficiently for developing concurrency-intensive systems only if there exist languages and tools corresponding to the described concepts. In this article there will be presented an approach based on using formalism. Theoretical advantages of formal specification are well known. However, usage of formal specification in practice ascertains some difficulties, thus their current advantages are not widely explored. The main focus of our research is to improve usage of formal method in verification of concurrency. Our vision consists in adapting the pragmatic approach and relaxing formalism, by creating graphical specification language based on events.

Keywords: software architecture, concurrency, formalization, specification, CSP#

1 Specification and formal methods

It is well known that methods of formal specification allow [2, 3]: to describe completely behavior of a system; to analyze in detail and, consequently, to better understand systems; to facilitate verification, maintenance and development of system; to reduce number of errors, etc.

In practice, the methods of formal specifications are not widely used due to the fact that formal character of specification languages makes difficult their understanding and usage.

Usually, overall development time using formal methods is expanded so much that the question arises whether to use them or not. The work [3] reflects unavailability of the majority of IT-experts to the methods of formal specifications and disadvantages of full formal analysis of large and complex systems. Despite this, the author of the work [3], being a proponent of formal specification methods, insists that the key to achieving good results is exactly the use of formal methods in early stages of development. Numerous studies [3, 4, 5] have shown that the earlier discrepancy is detected, the cheaper the error costs (Figure 1).

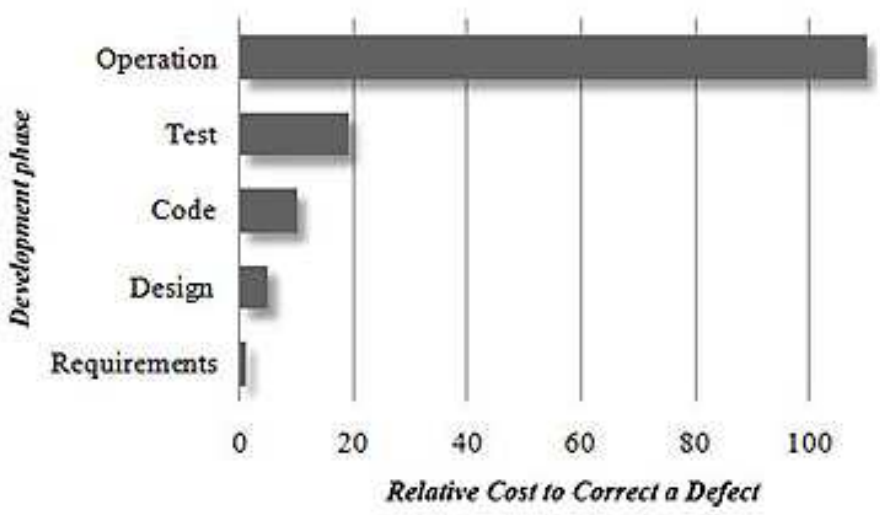


Figure 1. Relative cost to correct a defect

But is it always justified the effort for the formal specification of the system in the early stages of development? In [2] it is insisted that transition from informal requirements to formal specifications should not be done too early because of the fact that a greater degree of detail complicates the specification.

Also, in the above-mentioned work, it is proposed *a pragmatic ap-*

proach to formal specification, which consists in usage, if necessary, of several formalisms for specification of different system aspects, increasing expressiveness of formalization, in general, and avoiding restrictions of specification languages, in particular.

The same approach applies when using UML (Unified Modeling Language, [6]). However the language is semi-formal, even in the case of model annotating with OCL-expressions (Object Constraint Language, [7]). In addition to UML and ADL (Architecture Description Language) diagrams, at architectural level, for describing system behavior, the formal language needs to be used. Anyway, without doubt, today it is best used in the IT industry.

The expressiveness of UML models, the "standardization" of the software development processes, the advantages of the methods of formal specifications and pragmatic approach, incline to improve the specification phase through complement or extension of existing methods rather than through development of new universal and common language for formal specification.

2 Component-based structure architecting

Structure architecting consists in describing the set of components and set of connections (connectors) defining components interaction. Structure view of architecting process focuses on term, which is closely related to system topology.

Description of architecture as configurations of components and connectors is very popular, and, in majority of cases, architectural description languages (ADLs) graphically represent them as "boxes and lines".

The ACME language of architecture description must be mentioned. ACME is a formal meta-language, which provides a set of language constructs for describing architectural structure, architectural types and styles, and annotated properties of the architectural elements [8].

Publish-subscribe style architecture is represented in Figure 2 using ACME-like concepts and graphical representations.

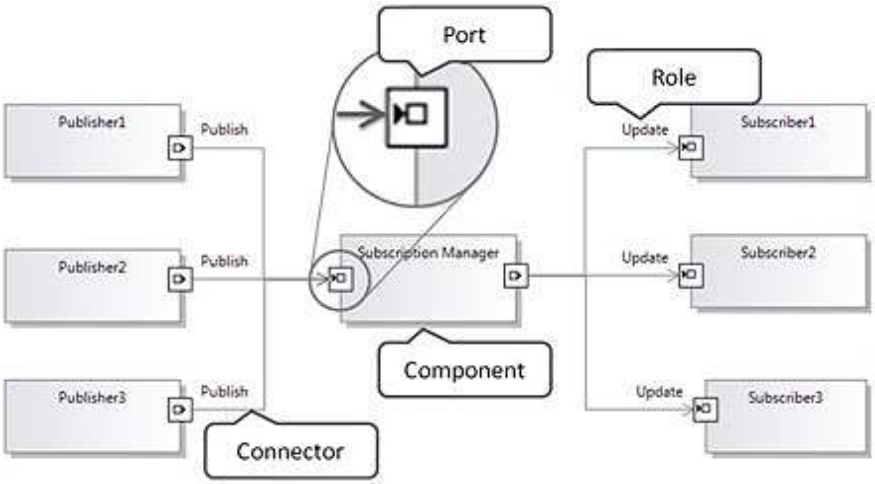


Figure 2. Structure view of a publish-subscribe architecture

Component is an elementary system entity, which represents process or data store unit of a system. Thus, components are abstractions used to model hardware and software elements. A module, library, class or other encapsulation unit can be an analogue of a component, abstract description of which can be captured by *properties*. Component functionalities are exposed through *ports*. Thus, port represents the interface which describes services (operations) of components that are provided or requested

Connector makes interaction as explicit concept. All architecture connectors embody a protocol of communication and synchronization between participants, which are defined by *roles*. Connectors have properties too. Details of interactions can be specified exactly using properties and another formalisms and tools as "value" (e.g. CSP-like language Wright or described bellow ESC language). Connectors and ports can be discoverable at run-time, and define a "transport" independent communication between components.

3 Event-based behavior architecting

3.1 Event centric development

Event-based behavior architecting permits very flexible interactions between system active entities, therefore event-driven techniques remain popular for concurrency for a long time. *Matt Welsh* [9] had adopted an event-driven architecture too, in order to support massive degrees of concurrency. Resulted *staged event-driven architecture* (SEDA) is very efficient and has a robust structure.

Event centric architecture in the simplest case defines four basic entities (Figure 3):

Figure 3. Basic entities of the event-based architecture

- Data (what data is processed);
- Actions (actions taken when processing);
- Components (where actions of processing localized);
- Events (when and in what sequence are actions activated).

Inexpensive synchronization is an important argument for event-driven approaches: synchronization is easily obtained by event cooperation [10]. Another key advantage of events is scheduling at application level. Thus scheduling optimization is possible. Also events allow better code locality, which is one of implementation mechanisms of the fundamental principle of modern software development – Separation of Concerns (SoC) [11, 12, 13].

3.2 Event-based Specification Chart (ESC)

The main objective of research is to simplify the specification of concurrency in information system architecting. If events happen in component ports and relationships among events are the only things that interest us, an event-based approach for behavior specification can be an adequate choice.

Event-based Specification Charts (ESCs) consist in drawings, which specify events, event orderings, event conflicts, roles and role actions.

An *event* is an instantaneous, atomic "state" transition in the computation trace. Exactly over these transitions the behavior of a system is defined. Event ordering in computation trace is determined by the *causal dependency* relationship "$\rightarrow$" (read as "*precede*"). The interpretation of "$\rightarrow$" as a causal ordering means that, if $e1$ and $e2$ are events in a system and $e1 \rightarrow e2$, then existence of the event $e1$ will cause occurrence of the event $e2$ in the future.

Conflict relationship "$\leftrightarrow$" models mutual exclusion of events, disallowing them to overlap in time. An *asymmetric conflict* "$\rightarrowtail$", which blocks an event while another event happens, is allowed too.

Thus an event e can occur when all its causes have occurred and no event, which it is in conflict with, has already occurred.

Concurrency and *non-determinism* are implicit in ESC specifications, and are determined by causal independence and absence of conflicts between concurrent events.

A *role* defines the behavior of a participant, identified in the collaborations between architecture components. Through roles the component captures events, which can occur in an order determined by

concurrency and synchronization logic, and exposes actions associated to these events by *role to cloud incidence* connectors.

Possibility of localization of interrelated events is realized using *cloud*. The cloud of events is a partially ordered set of events, where the partial ordering is determined by causal, temporal and other relations between events. A closely related term of event cloud is defined in the glossary of *Event Processing Technical Society* too [14].

Basics of the language notations are presented in Figure 4, where ESC chart describes the following entities:

- Events: $e1$, $e2$, $e3$, $e4$, $e5$;
- Causal dependencies: $e1 \rightarrow e2$, $e1 \rightarrow e3$, $e4 \rightarrow e5$;
- Conflict: $e3 \leftrightarrow e4$;
- Clouds: *Cloud1, Cloud2;*
- Role: *Role1.*

In order to increase flexibility of specifications two relationship attributes for causal dependencies were introduced: *role incidence* and *event occurrence*.

Role incidence attribute is used to indicate which roles are implicated in event occurrence, and has the following values:

- *Same* – the causal and dependent events occur involving the same role;
- *All* – a special case of *same* attribute, used when a dependent event occurs only if the causal event is occurred in every incident role. Thus the cause event is synchronization between all component processes interfaced by roles.
- *Any* – a dependent event occurs, when the causal event manifested itself in any incident role; used as attribute for dependencies between events located in different clouds;

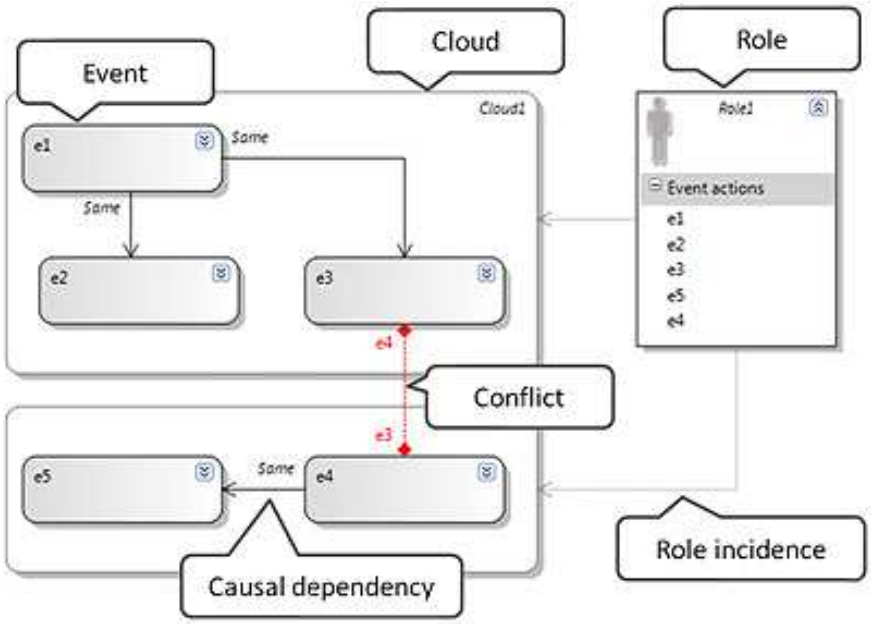


Figure 4. An ESC diagram

Event occurrence attribute can be used to specify how many times the dependent event may occur, when the causal event manifests once:

- *bound[n]* – limits to n number of occurrences of the dependent event in each incident roles;
- *free* – the dependent event may occur unlimited times for incident roles.

Let there be four events ordered by some dependencies ($e1 \rightarrow e2$, $e2 \rightarrow e3$, and $e2 \rightarrow e4$), which form a system behavior presented in Figure 5. *Role1* and *Role2* are synchronized by event $e1$, because dependency between $e1$ and $e2$ has *All* as value of *role incidence* attribute. Event $e2$ thus will not happen until $e1$ will not occur in both roles. Also according to diagram from Figure 5, the event $e3$ will occur only once

for each incident role (both *Role3* and *Role4*), but the event $e4$ will occur unlimited times involving incident roles.

It is important to mention that role can be "interpreted" by a process or multiple. Thus event $e3$ may occur once in multiple processes or will occur multiple times in one process (execution unit).

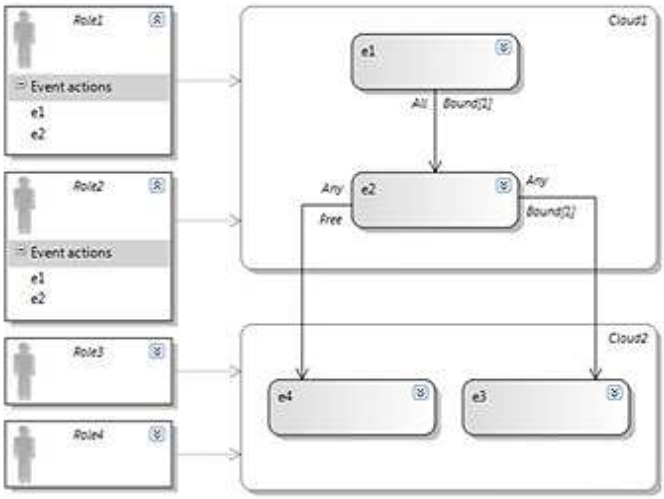


Figure 5. Synchronized events in ESC

Modern software development requires an incremental refinement approach. Thus reutilization is an important factor to efficient elaboration, but it is applied to late. Reutilization, usually achieved through inheritance, is especially present from design to programming activities. Therefore an effort must be made to apply reusing in behavioral specification of concurrency-intensive architecture too.

Refinements in ESC language are realized through *cloud superposition* (with *event specialization*) and *event substitution*.

Cloud superposition superimposes additional behavior on an existing cloud. Addition behavior preserves independencies and conflicts between old events, therefore *liveness* and *safety* properties will be preserved. Superposition must be viewed as *monotonic inheritance*,

because can be characterized in the following way: new events may be added; new actions may be associated to new events; new causal dependencies may be added between new events, new causal dependencies may be added between a specialized event and events from superpositioned cloud; a specialized event may be refined through *substitution* by events from a cloud.

Event substitution is a refinement, which replaces an event by multiple events from a cloud. Substitutions preserve dependencies of replaced events, therefore causal ordering is preserving too.

Refinement relations between clouds and events is shown in Figure 6, according to which *Role1* is involved by events $e1$, $e3$, $e4$, $e5$, $e7$ with occurring ordering determined by the following final (after refinements) causal dependencies: $e1 \rightarrow e3$, $e1 \rightarrow e4$, $e1 \rightarrow e5$, $e4 \rightarrow e7$, $e5 \rightarrow e7$.

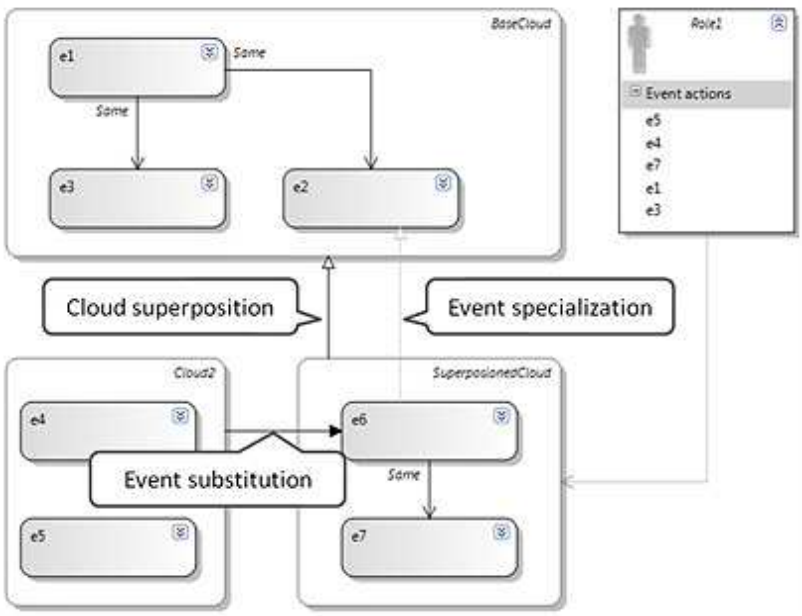


Figure 6. Superposition and substitution refinements

3.3 An event structure semantic for ESCs

Every specification language should have a formal semantic, which defines logic for reasoning about behavior and concurrency. A semantic is indispensable for the development of tools; therefore it is an advantage, if specification language has direct and simple correspondence to the logic. This is the way to avoid errors in concurrent specification.

Another important aspect of language design concerns explicit linking of specification to implementation, which is realized at later stages of developments. So language must be capable to preserve internal system "structure" (which can be lost with interleaving models [15]), to express implicitly non-determinism (which is an inherent property of modern multi-threading programs [16]), and to avoid detailed internal state description (behavioral approach is preferred for high-level architectural specification).

According to [15, 17] and aforementioned requirements, a well accepted branching-time true concurrency semantic model is model of event structures (Figure 7).

Figure 7. The ESC language as an emanation of branching-time true concurrency extensional semantic

An *event structures* model with non-interleaving semantic allows in

a natural way to describe the relationship between events of the system [15, 17].

The event structure $S = (E, \leq, \#)$ consists of a countable set of events E, partially ordered by causal dependence $\leq \ \subseteq (E \times E)$, an irreflexive and symmetric conflict $\# \subseteq (E \times E)$, satisfying the principle of inheritance: $\forall e, e1, e2 \in E;\ e\#e1 \leq e2 \Rightarrow e\#e2$. Thus the concurrency relationship *co* between events e and $e1$ from E can be defined as follows: e *co* $e1$, iff $\neg(e1 \leq e \vee e \leq e1 \vee e\#e1)$.

Event substitution refinement can be defined as *vertex substitution* operation presented in [18] with assumption that substituted event has no conflict relationship with any events (Figure 8).

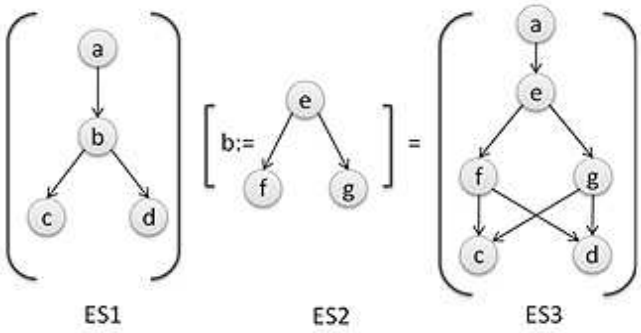


Figure 8. Event substitution

In order to describe formally this refinement operation it needs to assume that

- $U(E_1, e_1)$ is a subset of E_1 upper bounded by $e_1 \in E_1$ ($\forall u \in E_1$: $u \leq e_1$);
- $L(E_1, e_1)$ is a subset of E_1 lower bounded by $e_1 \in E_1$ ($\forall l \in E_1$: $e_1 \leq l$);
- $\lceil S \rceil$ denotes a subset of S, including only elements' upper bound S ($\forall s \in S;\ \forall u \in \lceil S \rceil;\ s \leq u$);

- $\lfloor S \rfloor$ denotes a subset of S, including only elements' lower bound S ($\forall s \in S;\ \forall l \in \lfloor S \rfloor;\ l \leq s$).

Then after substitution of event b in event structure ES1 by event structure ES2 (operation which can be noted as in [19] with ES3 = ES1[$b \rightarrow$ES2]) is obtained an event structure ES3=$(E_3, \leq_3, \#_3)$, where

- $E_3 = (E_1 \cup E_2) - \{b\},\ E_1 \cap E_2 = \emptyset$;
- $\leq_3 = (\leq_1 - \leq_b) \cup \leq_2 \cup \leq_{\lceil U(E1,b) \rceil \times \lfloor E2 \rfloor} \cup \leq_{\lfloor L(E1,b) \rfloor \times \lceil E2 \rceil}$;
- $\#_3 = \#_1 \cup \#_2$.

Refinement through *cloud superposition* formally can be interpreted as union of two graphs, which anticipates vertex contractions specified by specialization relationships (dashed arrow in Figure 9).

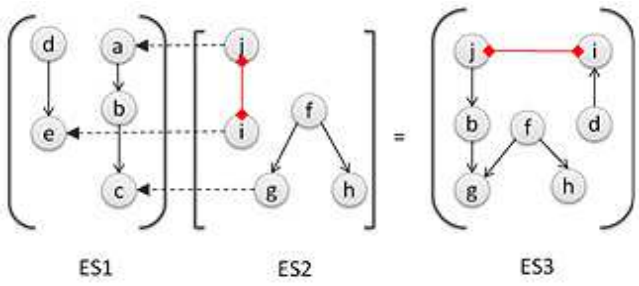


Figure 9. Cloud superposition

The above-described basic relationships are the ones which permit event structures to be expressive and natural for concurrency description and explaining. However these arguments are not valid for automated analysis, because of the difficulty in defining operations like products and parallel compositions on event structures of the form $(E \leq, \#)$. They encourage the uses of more general structures or even CCS/CSP like languages from which event structure semantics is then induced [15].

3.4 Operational interpretation of ESC specifications

Automated analyses imply an operational interpretation of formal ESC specifications. The most appropriate for this is the classic process algebra CSP (*Communicating Sequential Processes*). CSP has an event-based operational semantic and describes system behavior by sequencing of events [20]. There are two *popular* model checkers based on CSP-like operational semantic and syntax: FDR (Failures/Divergence Refinement), which is a commercial product expressing models in machine-readable dialect of CSP (CSP_M) [21]; and PAT (Process Analysis Toolkit) with CSP# as input language [22].

Greater flexibility, expressiveness, openness and freeness offered by PAT determine a univocal choice for CSP# as interpretation language. Compared to CSP_M, CSP# adds to original language features such as shared variables, asynchronous communication channels and event associated programs [23].

Formally, the language of sequential processes (programs) CSP#, ranged over processes P and Q, is the set of terms generated by the following BNF *(Backus-Naur Form)* description (Figure 10), in which e is a name representing an event with an optional sequential program prog, X is a set of event names, b is a Boolean expression,ch is a channel, exp is an expression, and x is a variable.

```
P ::= Stop | Skip                                          (event prefixing)
  | e{prog} -> P                                           (channel operations)
  | ch!exp -> P | ch?x -> P                                (hiding event and interleaving)
  | P \ X | P ||| Q                                        (sequential and parallel composition)
  | P;Q | P || Q                                           (external and internal choices)
  | P[]Q | P<>Q                                            (conditional choice)
  | if b {P} else {Q}                                      (guarded process)
  | [b]P
  | P interrupt Q
```

Figure 10. The BNF description of CSP# language

The easiest way to apply PAT platform as model checker is to create a "rewriter" from the ESC specification language to CSP# language. In

order to avoid complex CSP# models, rewriting should be anticipated by refinement operations, which can be understood as operations of syntactic substitution.

Rewriting from ESC model to CSP# model produces systems composed of multiple *processes*. In accordance with CSP# language definition the word *process* can stand for the behavior pattern of an object, which can be described in terms of limited set of events. In both languages each event name denotes an event class; there may be many occurrences of the same event, involved separately in time by processes (CSP#) or roles (ESC).

Let *Role1* and *Role2* be the roles, which act and interact with each other exactly in accordance with ESC specifications showed in Figure 11. It is easy to see, that *Role1* exposes actions for occurrences of events a, b and c, ordered by causal dependencies from *TheCloud1*; and the behavior of *Role2* is determined by ordering of occurrences of events a, b, c (from *TheCloud1)*, and d, e, f (from *TheCloud2*). Also according to specification, event b from *TheCloud1* requires simultaneous participation of the both roles involved.

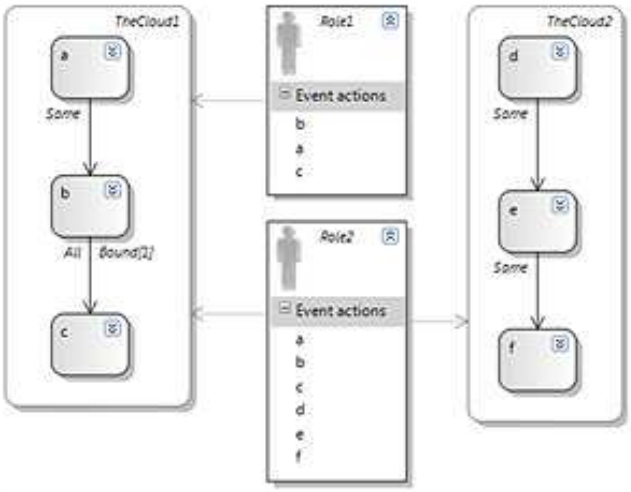


Figure 11. Event clouds as behavior patterns

The system behavior is defined by all role behaviors, which can be described as combination of deterministic CSP processes. When roles are incident to the same cloud, parallel composition is used. *Parallel composition* introduces concurrency and synchronization. Using of the same alphabet assures that processes interact in lock-step synchronization over events with the same name. *Interleaving* operator may combine many role behaviors (incident to different event cloud) into complex one, representing completely concurrent activity.

Thus a CSP# model which describes a behavior equivalent to ESC specification from Figure 11 consists from five processes (Figure 12): a composed process *System*, which corresponds to overall specified system; processes $R1$ and $R2$, which correspond to roles *Role1* and *Role2*; processes $R21$ and $R22$, which combine concurrently behavior of *Role2* determined by *TheCloud1* and *TheCloud2* from ESC specification.

```
R1 = a1 -> b -> c1 -> R1;
R21 = a2 -> b -> c2 -> R21;
R22 = d2 -> e2 -> f2 -> R22;
R2 = R21 ||| R22;

System = R1 || R2;
```

Figure 12. Interleaving and parallel composition in CSP# model

Also parallel composition is used to define a behavior like presented in Figure 13, according to which *Role1* may concurrently involve events (b and d, b and e, c and e).

After model "rewriting", the role process $R1$ behaves like the system composed of processes $R11$, $R12$ and $R13$, interacting in lock-step synchronization as described above (Figure 14).

Last translations may not be optimal, because the processes describing the behavior *Role1* are obtained by finding in the cloud's DAG all the paths that begin from source events and end at the sink events. The equivalence of ESC model and CSP# model can be proved, if both models satisfy the same state graph (Figure 15). This equivalence is

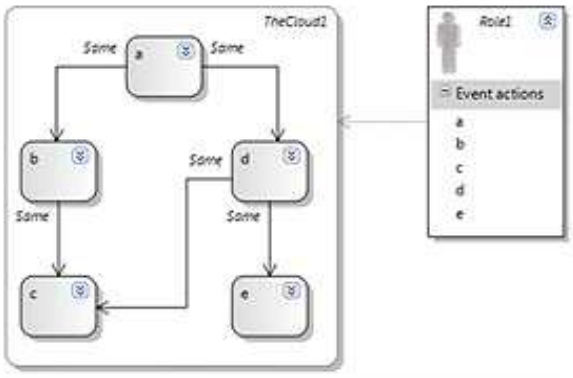


Figure 13. A cloud with concurrent events

```
R11 = a -> b -> c -> R11;
R12 = a -> d -> c -> R12;
R13 = a -> d -> e -> R13;

R1 = R11 || R12 || R13;
```

Figure 14. One cloud can impose multiple processes for behavior definition

analogical to Milner's principle of observational congruence [20].

It is easy to see from the state graph (Figure 15) that two independent events (c and e, d and b, b and e) which can occur at the same state, can be performed in any order without affecting the reached state (in literature this is often identified as 'diamonds' of concurrency [17]).

In Figure 16 represented role behaviors are defined by two clouds, which include conflicting events. Conflict relationship is used to specify mutual exclusion between events. Inheritance principle of conflict relationship acts in cloud boundary. Circle in the upper-left corner of cloud shape means that the cloud is a singleton: at the same time only one role incident to cloud can involve events from this cloud.

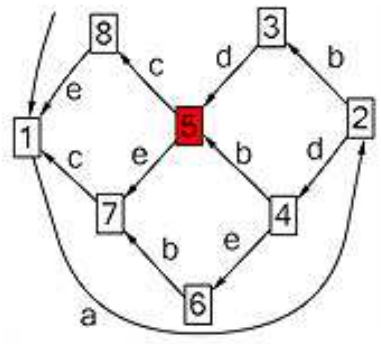


Figure 15. Complete state graph (generated by PAT)

It is obvious that conflict relationship imposes a choice between processes including conflicting events. Selection in CSP model will be done in the *conflict process* specifically designed for this. Conflict process must be defined by role events. Deterministic choice realized in CSP by a special operator – *external choice,* can help us to engage in execution only one "conflicting" process. Thus specification shown in Figure 16 can be translated as presented in Figure 17.

In the system described in Figure 17, the lock-step synchronization admits execution of one process at a moment. Using global variables and guarded processes, and splitting alternative process in two (sub) processes can permit occurrence of one conflicting event in multiple role processes (Figure 18).

4 Conclusion

The Event-based Specification Charts (ESC) specification language was created to easily design model of concurrent processes using events. Thus, the proposed language involving event-driven techniques, does not replace the existing verification methods and tools. On the contrary, the main objective of the language is to improve usage of formal methods in verification.

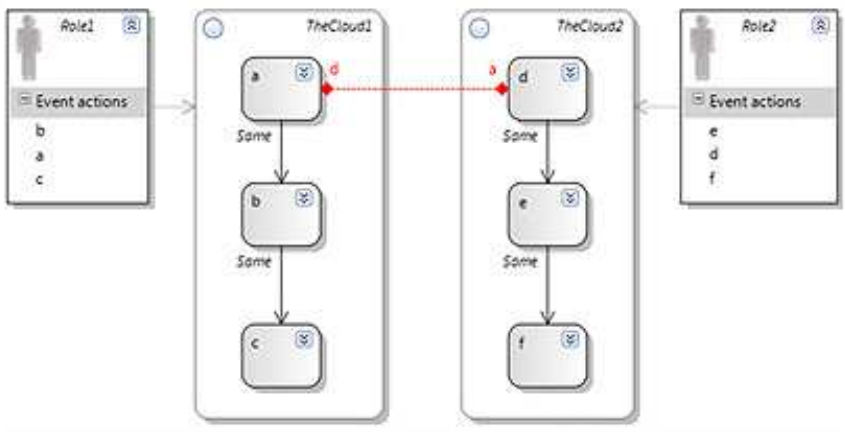


Figure 16. Conflict between events from singleton clouds

```
R1 = a -> b -> c -> R1;
R2 = d -> e -> f -> R2;
Conflict = a -> c -> Conflict [*] d -> f -> Conflict;

System = Conflict || (R1 ||| R2);
```

Figure 17. Conflict relationship and external choice

Event-based specification charts have many advantages. They are easily understood, even by the non-specialist. They permit to concentrate on causality or independence of events. They offer reutilization mechanisms of events.

Language constructs semantically are interpreted by means of event structure model, which is more suitable for modeling "true parallelism". Hoare's CSP language provides operational interpretation for ESC models: partial order is modeled by the sequencing; conflict relation is modeled by nondeterministic choice, etc.

Major factor reducing the use of formal language is poor integration of formal tools with modern development environments. Now we intend to develop a tool for specification of concurrency, integrated into Visual

```
Var cr1 = 0;
var cr2 = 0;

R1() = a -> b -> c -> R1();
R2() = d -> e -> f -> R2();
Cf() = [cr2 == 0] a -> {cr1++} -> Cf() []
   [cr1 > 0] c -> {cr1--} -> Cf() []
   [cr1 == 0] d -> {cr2++} -> Cf() []
   [cr2 > 0] f -> {cr2--} -> Cf();

System() = Cf() || (|||x:{1..2}@(R1() ||| R2()));
```

Figure 18. Multiple role processes with conflicting events

Studio IDE, using Visual Studio Visualization and Modeling SDK, in order to increase the quality of software development. Formal checks will be made by PAT, which provides an API just for this.

References

[1] D. Ciorba, V. Besliu. *Architecting software concurrency.* Computer Science Journal of Moldova. Chisinau : s.n., 2011. Vol. 19, 1(55), pp. 92–108. http://www.math.md/files/csjm/v19-n1/v19-n1-(pp92-108).pdf. ISSN 1561-4042.

[2] M. Heisel. *A pragmatic approach to formal specification.* [ed.] H. Kilov and W. Harvey. Object-oriented behavioral specification. Boston / Dordrecht / London : Kluwer Academic Publishers, 2006.

[3] A. Hall. *Realising the Benefits of Formal Methods.* Journal of Universal Computer Science. 2007, Vol. 13, 5.

[4] LKP Consulting Group. *The Real Cost of Software Defects.* [http://www.lkpgroup.com/Cost%20of%20Software%20Defects.pdf] Atlanta, US : s.n., May 2006.

[5] Intel Corporation. *Intel® Fast Track Initiative: New Tools for Enabling the Latest Technologies.* [http://www.intel.com/partner/ft_webinar/Intel-FastTrack-SAT-Whitepaper_FINAL_4-14-2010.pdf] 2010.

[6] Object Management Group Inc. *UML 2.3. The Object Management Group.* [Online] May 9, 2010. [Cited: Nov 21, 2010.] http://www.omg.org/spec/UML/2.3/.

[7] Object Management Group Inc. *Object Constraint Language (OCL).* [Online] Feb 2010. [Cited: Nov 21, 2010.] http://www.omg.org/spec/OCL/.

[8] A. Kompanek. *Modeling a System with Acme.* [http://www.cs.cmu.edu/∼acme/html/WORKING-%20Modeling%20a%20System%20with%20Acme.html] s.l. : Carnegie Mellon University, 1998.

[9] M. Welsh. *An Architecture for Highly Concurrent, Well-Conditioned Internet Services.* Ph.D. Thesis. Berkeley : University of California, August 2002.

[10] M. Fowler. *Event Collaboration.* Development of Further Patterns of Enterprise Application Architecture. [Online] Jun 19, 2006. [Cited: Dec 05, 2010.] http://martinfowler.com/eaaDev/EventCollaboration.html.

[11] W. Hürsch, C. Lopes. *Separation of Concerns.* College of Computer Science, Northeastern University. Boston, USA : s.n., 1995. Technical report.

[12] C. Constantinides, T. Elrad. *On the Requirements for Concurrent Software Architectures to Support Advanced Separation of Concerns.* OOPSLA'2000, Workshop on Advanced Separation of Concerns in Object-Oriented Systems. 2000.

[13] G. Kiczales, et al. *Aspect-Oriented Programming.* Proceedings of ECOOP'97. s.l.: Springer-Verlag, 1997.

[14] Event Processing Technical Society. *Event Processing Glossary.* [http://www.ep-ts.com/] [ed.] D. Luckham and R. Schulte. July 2008.

[15] G. Winskel, M., Nielsen. *Models for Concurrency.* [http://www.daimi.au.dk/PB/463/PB-463.pdf] November 1993. DAIMI PB-463.

[16] R. Carver, K. Tai. *Modern multithreading: implementing, testing, and debugging multithreaded Java and C++/Pthreads/Win32 programs.* [http://books.google.ro/books?id=Wuex4orZgOEC] s.l.: John Wiley and Son, 2006. ISBN 0-471-72504-8.

[17] V. Sassone, M. Nielsen, G, Winskel. *Models for Concurrency: Towards a Classification.* Theoretical Computer Science. 1996. Vol. 170, 1-2, pp. 297–348.

[18] T. Basten. *Parsing Partially Ordered Multisets.* International Journal of Foundations of Computer Science. s.l. : World Scientific Publishing Company, December 1997. Vol. 8, 4, pp. 379–407.

[19] U. Goltz, R. Gorrieri, A. Rensink. *Comparing Syntactic and Semantic Action Refinement.* Information and computation. s.l.: Academic Press, Inc., 1996. 125, pp. 118–143.

[20] C. Hoare. *Communicating Sequential Processes.* [http://www.usingcsp.com/] s.l.: Prentice Hall International, 2004.

[21] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement. FDR2 User Manual.* [http://www.fsel.com/fdr2_manual.html]. Octomber 2010.

[22] Y. Liu, J. Sun, J. Dong. *Developing Model Checkers Using PAT.* [ed.] A. Bouajjani and W. Chin. Automated Technology for Verification and Analysis – 8th International Symposium, ATVA 2010. Singapore : Springer, September 2010. Vol. 6252, pp. 371–377.

[23] J. Sun, et al. *Integrating Specification and Programs for System Modeling and Verification.* [ed.] W. Chin and S. Qin. Proceedings of the third IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE'09). s.l.: IEEE Computer Society, 2009. pp. 127–135.

Dumitru Ciorbă, Victor Beşliu

Received May 28, 2011

Technical University of Moldova
Automation and Information Technology Department
Str. Studenţilor, 7/3, corp 3, 504 Chişinău, MD-2068
Phone: (+373 22) 509908
E–mail: *victor.besliu@ati.utm.md*, *besliu@mail.utm.md*, *vbesliu@yahoo.com*
E–mail: *diciorba@yahoo.com*, *dumitru.ciorba@ati.utm.md*

Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

Concept-Oriented Model: Extending Objects with Identity, Hierarchies and Semantics

Alexandr Savinov

Abstract

The concept-oriented data model (COM) is an emerging approach to data modeling which is based on three novel principles: duality, inclusion and order. These three structural principles provide a basis for modeling domain-specific identities, object hierarchies and data semantics. In this paper these core principles of COM are presented from the point of view of object data models (ODM). We describe the main data modeling construct, called concept, as well as two relations in which it participates: inclusion and partial order. Concepts generalize conventional classes by extending them with identity class. Inclusion relation generalizes inheritance by making objects elements of a hierarchy. We discuss what partial order is needed for and how it is used to solve typical data analysis tasks like logical navigation, multidimensional analysis and reasoning about data.

Keywords: Data modeling, object data models, set nesting, partial order, data semantics.

1 Introduction

The concept-oriented model (COM) is an emerging general-purpose approach to data modeling. It is aimed at unifying different views on data and solving a wide spectrum of problems in data modeling and analysis [31, 33]. COM overlaps with many existing data modeling methodologies but perhaps most of its features are shared with object data models (ODM) [10, 4, 3]. COM is not only based on the general principles of object-based models but can be viewed as their further

development and generalization. If we take ODM as a starting point then what problems COM is going to solve? In other words, what are the main motivating factors behind this approach if it is considered a generalization of ODM? COM is aimed at solving the following three major problems:

How objects exist. COM provides a mechanism for *domain-specific references* so that *both* objects and their references may have arbitrary structure and behavior.

Where objects exist. COM turns each object into a set which is interpreted as a space, context, scope or domain for its member objects. The whole model is then turned into a *set-based* approach where sets are first-class elements of the model.

What objects mean. COM augments objects with *semantics* and makes references elementary semantic units. The meaning of an object is defined via other objects and this semantic information can be used for reasoning about data.

"Object identity is a pillar of object orientation" [16] and the role of identities has never been underestimated. There exist numerous studies [17, 38, 1, 16, 11] highlighting them as an essential part of database systems and arguing for the need in having strong and consistent notion of identity in data and programming models. A lot of identification methods have been proposed like primary keys [6], object identifiers [1, 17], l-values [18] or surrogates [12, 7]. Nevertheless, there is a very strong bias towards modeling entities while the support of identities is relatively weak. Identities have always been considered important but secondary elements remaining in the shadow of their more important counterpart. There is a very old and very strong belief that it is entity that should be in the focus of data modeling while identities simply serve entities. Almost all existing data (and programming) models assume that identities should be provided by the platform and there is no need for modeling domain-specific identities. For this reason, the roles of entities and identities are principally separated: entities have

domain-specific structure and behavior while identities have *platform-specific* structure and behavior.

COM changes this traditional and currently dominating view by assuming that identities and entities are equally important for data modeling. In this context, the main goal of COM is to provide data modeling means where *both* identities and entities may have arbitrary domain-specific structure and behavior. To solve this problem, COM introduces a novel data modeling construct, called *concept* (hence the name of the model), which generalizes classes. Its main advantage is that it allows for modeling arbitrary domain-specific identities (references) what is impossible if conventional classes are used. Importantly, identities and entities are modeled together as two parts of one thing. Elements in COM can be compared to complex numbers in mathematics which also have two constituents (real and imaginary parts) manipulated as one whole.

Set-orientation is one of the most important features of any data model just because data is normally represented and manipulated as groups rather than as individual instances. Probably, it is the solid support of set-oriented operations why the relational model [6] has been dominating among other data models for several decades. And insufficient support of the set-oriented view is why object-oriented paradigm is less popular in data modeling than in programming. Of course, we can model sets, groups or collections manually (at conceptual level) but in this case the database management system is unaware of these constructs and cannot help in maintaining consistency of these structures. In this context, the main goal of COM is to turn instance-based view into a set-based model where set is a first-class notion supported by the model at the very basic level. The idea here is that any instance has to be inherently a set, and vice versa, a set has to be a normal instance. To solve this problem, COM introduces a novel relation, called *inclusion*. The most important change is that elements exist within a hierarchy where any parent is a *set* of its children and any instance is a member within its parent superset. Parent elements are shared parts of children and are treated as a space, context, scope or domain for its children. In contrast, in most class-based models classes exist within

a hierarchy while their instances exist in flat space (so we do not have a hierarchy of instances). The use of inclusion relation makes COM similar to the hierarchical model [37] where parents are containers for their child elements. In addition, inclusion generalizes inheritance and can be used for reuse, type modeling and other purposes. This approach is also very close to prototype-based languages [5, 19, 34] where parents are shared parts of children. However, these languages do not use classes while COM allows for using both classes (in the form of concepts) and object hierarchies (in the form of inclusion).

According to Stefano Ceri [20], "the three most important problems in Databases used to be Performance, Performance and Performance; in the future, the three most important problems will be Semantics, Semantics and Semantics". The main advantage of having semantics in databases is that it "should enable it to respond to queries and other transactions in a more intelligent manner" [7] by providing richer mechanisms and constructs for structuring data and representing complex application-specific concepts and relationships. There has been a tremendous interest in semantic models [13, 21] but most of the works propose conceptual models which need to be translated into some logical model. The lack of semantics in logical data models significantly decreases their overall value and, particularly, decreases the possibility of information exchange, data integration, consistency and interoperability.

As a response to this demand, COM proposes a novel approach to representing and manipulating semantics as integral part of the model. The main idea is that concepts are partially ordered by using the rule that a field type represents a greater concept. As a result, all instances exist as elements of a *partially ordered set* where they have greater elements and lesser elements. In contrast, most other models are graph-based with objects treated as nodes and references considered the edges. In COM, references always represent greater elements by playing a role of an elementary semantic construct rather than a navigational tool in a graph. Partial order changes the way how data is being accessed: instead of using graph navigation, COM introduces two operations of *projection* and *de-projection*. Although there exist approaches which

are based on partial order relation [24], only in COM it is used as a primary relation for representing data semantics and reasoning about data. The main benefit of using partial order is that it "seems to fulfill a basic requirement of a general-purpose data model: wide applicability" [24], that is, many conventional data modeling mechanisms and patterns can be explained in terms of this formal setting. In COM, this approach was shown to be successful in solving such highly general tasks as logical navigation [27], multi-dimensional analysis [26], grouping and aggregation [28], constraint propagation and inference [29].

In the next three sections we discuss three principles of COM. In Section 2 we describe duality principle and introduce the main data modeling construct, concept. In Section 3 we discuss inclusion principle by defining inclusion relation among concepts and showing its differences from inheritance. Section 4 discusses order principle by demonstrating how partial order can be used for data access and solving typical data modeling tasks. Section 5 makes concluding remarks. We will follow a convention that identities are shown as gray rounded rectangles while entities are white rectangles. Also, concepts names will be written in singular while collection names will be in plural.

2 Concepts Instead of Classes

Existence precedes and rules essence — Jean-Paul Sartre, Being and Nothingness

2.1 Modeling Identities

How things exist and what does it mean for a thing to exist? In many theoretical and practical contexts existence can be associated with the notions of *representation* and *access*. This means that a thing is assumed to exist if it can be represented and accessed. Conversely, if a thing does not have a representation or cannot be accessed then it is assumed to be non-existing. According to this view, any existing thing is supposed to have a unique *identity* which manifests the fact of its existence.

One of the main distinguishing features of COM is that it makes

identities first-class elements of the model. To describe how COM differs from other identification schemes we will use the following three dimensions:

- Strong vs. weak identities
- Domain-specific vs. platform-specific identities
- Implicit vs. explicit identities

The first criterion divides all identification schemes in two groups [38, 11]: *strong identities* and *weak identities*. Strong identities exist separately from the identified entity while weak identities are actually internal properties of the entity used for identification purposes (identifier properties).

Memory address is an example of a strong identity because memory cells do not contain their address as a property (otherwise memory would be a two-column array with addresses in the first columns and cells in the second column). Object references are also strong identities because they are not contained in object fields. In particular, if a class does not have fields then its instances have zero size but still have references. In contrast, primary keys in the relational model (RM) are weak identities because they are made up of a number of normal attributes.

The main problem with weak identities like primary keys is that it is not possible to access entity properties without some kind of strong identity which therefore should be provided by a good data model. In other words, some kind of strong identity must always exist while having weak identities in the model is optional. In this sense, weak identities should be viewed as a design pattern rather than a primary data modeling mechanism.

Strong identities are provided in ODM and COM because objects in these models are represented by a separable reference (Fig. 1). In contrast, RM relies on weak identities in the form of primary keys. Note however that many implementations provide some kind of strong identity for representing rows like surrogates and their usefulness is theoretically justified.

The second dimension for evaluating various identification schemes

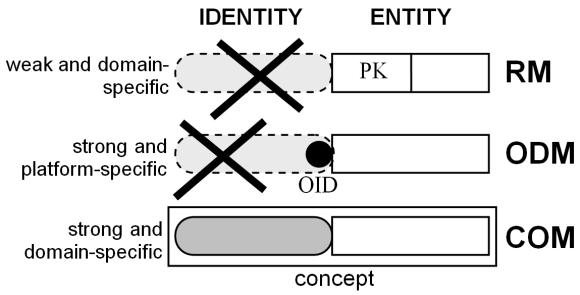


Figure 1. Modeling identities and entities in different models

is the possibility to model *domain-specific* (user-defined) identities as opposed to having only *platform-specific* (primitive, system) identities. An example of platform-specific identity is OID, surrogate or object reference, which are provided by the DBMS. The mechanism of primary keys is an example of domain-specific identities which is defined by the data modeler and reflects specific features of the problem domain. Since identities are integral part of the problem domain, a good data model should provide an adequate mechanism for their modeling. In particular, this mechanism is available in RM and COM but not in ODM (Fig. 1).

The third criterion separates all identification mechanisms depending on whether they provide *implicit* or *explicit* identities. An identity is called explicit if its functionality has to be used manually to access the represented entity. For example, primary keys are explicit identities because it is necessary to manually specify join conditions. The problem of explicit identities is that one and the same fragments span many queries. If the structure of the identity changes then all queries involving this fragment have to be also updated. For example, all queries where we need to get bank accounts given a set of persons involve the same fragment with the join condition like the following:

```
WHERE ACCOUNTS.owner = PERSONS.id AND ...
```

Note that this fragment is mixed with other conditions. What is worse,

if we change the primary key then all queries involving it in their join conditions have to be also updated (join is a cross-cutting concern).

In contrast, implicit identities hide their structure and functionality. For example, each time an object is about to be accessed, the DBMS needs to resolve its OID, then to lock memory handle and finally execute the operation using the obtained memory address. If it is a remote reference then the access procedure is even more complex. However, all these operations in queries are hidden – it is enough only to specify a variable and operation to be executed – all the intermediate actions will be executed behind the scenes. The mechanism of implicit identities has numerous advantages and therefore it should be supported by good data models. However, dot notation is also rather restrictive in comparison with the flexibility of joins. In COM and ODM, implicit identities are supported at basic level while RM relies on explicit identities in the form of join conditions for FK-PK pairs.

2.2 Identity and Entity — Two Sides of One Thing

To provide strong domain-specific implicit identities COM assumes in its duality principle that *any element is a couple consisting of one identity and one entity*. Formally, an element is represented as a couple of one identity tuple and one entity tuple. To model such identity-entity couples COM introduces a novel construct, called *concept*. Concept is defined as a couple of two classes: one identity class and one entity class. Both classes may have fields which are referred to as *dimensions* (to emphasize their special role for defining partial order described in Section 4) as well as other members like methods and queries. Importantly, these two constituents cannot be used separately because identity-entity couples are regarded as one whole. When a concept is instantiated we get a couple consisting of one identity and one entity so that the whole approach is reduced to manipulating identity-entity couples. Identity is an observable part manipulated directly in its original form. It is passed by-value (by-copy) and does not have its own separate representative. Entity can be viewed as a thing-in-itself or reality which is radically unknowable and not observable in its original

form. The only way to do something with an entity consists in using its identity as an intermediate. Identities are transient *values* while entities are persistent *objects*.

For example, a bank account in COM can be described by the following concept:

```
CONCEPT Account
   IDENTITY
      CHAR(10) accNo
   ENTITY
      DOUBLE balance
```

Identity class of this concept has one dimension which stores account number and entity class has one dimension which stores the current balance.

Concepts generalize conventional classes and are used instead of them in COM. There are two special cases:

- identity class is empty
- entity class empty

If identity class is empty then such concept is equivalent to a conventional class. Its instances will be represented by identities inherited from the parent concept (see Section 3). For example, we could define a class of color objects as follows:

```
CONCEPT ColorObject
   IDENTITY // Empty
   ENTITY
      INT red, green, blue
```

Instances of this concept are normal objects represented by some kind of platform-specific reference or OID. If all concepts have empty identity classes then we get the object-oriented case where one and the same platform-specific identity is inherited from the root concept and is used to represent *all* entities. What is new in COM is that it allows for modeling user-defined types of references together with the represented entities. Such references can be thought of as user-defined surrogates or separable primary keys. If entity class is empty then this concept

describes a value type (value domain). Indeed, its identity does not have entity part and hence it is not intended to represent anything (if it is not used as identity in a child concept). For example, we could define colors as values:

```
CONCEPT ColorValue
   IDENTITY
      INT red, green, blue
   ENTITY // Empty
```

Any instance of this concept is a value which is copied when it is passed or stored.

Concepts are instantiated precisely as classes by specifying their name as a type of the variable and then invoking this concept constructor. Moreover, from the source code where concepts are used, we cannot determine if it is a concept-oriented query or an object-oriented one – it depends how the concepts are defined. If they have only one constituent (entity) then it is an object-based approach and if they have two constituents (identity and entity) then it is a concept-oriented approach.

In data modeling concepts are normally used to declare types of elements in collections where data are stored. For example, if data are stored in tables then concepts are used to parameterize such a table by specifying the type of its elements. In an SQL-like language a table for storing bank accounts could be created as follows:

```
CREATE TABLE Accounts CONCEPT Account
```

Here `Accounts` is a table name and `Account` is a concept name so that this new table will store only elements of this concept type. Note that collections in COM are different from relational tables because they have domain-specific row identifiers. Such a collection can be viewed as a memory with arbitrary user-defined addresses and arbitrary user-defined cells.

2.3 Concepts for Domain Modeling

COM provides means for describing both values (identity class) and objects (entity class). The specific feature is that they are described and

then exist in couples. In contrast to object-relational models (ORM) [35, 36] where value domains and relations are modeled separately, COM provides one unified construct for modeling simultaneously value and relation types. In other words, there is only one type – a type of an element – but this element has a transient part (value) and a persistent part (object). The main benefit is that we can vary the "degree of persistence" by choosing which attributes belong to transient part (identity) and persistent part (entity).

In ORM, the idea is that relation attributes can store complex values rather than only primitive values. For example, we can define a user defined type composed of two integers:

```
TYPE MyType
   field1 AS INT
   field2 AS DOUBLE
END TYPE
```

After that it can be used as a type of relation attributes:

```
RELATION MyRelation
   intAttribute AS INT
   myAttribute AS MyType
END RELATION
```

The problem is that traditional approaches to data and type modeling do not allow us to use relations as types:

```
TYPE NewType
   myField AS MyRelation // Not possible
END TYPE

RELATION NewRelation
   myAttribute AS MyRelation // Not possible
END RELATION
```

In other words, relations are defined on domains only (primitive or complex) but not on other relations.

COM solves this problem by using concepts which essentially unite domain modeling and relation modeling. In COM, there is only one kind of domain or set – it is a set of concept instances which are value-object couples. Once a concept has been defined, it can be then used as

a type in any other concept independent of its use for value modeling, relation modeling or mixed use. For example, we could define colors as values:

```
CONCEPT Color
   IDENTITY
      INT red, green, blue
   ENTITY // Empty
```

This concept is equivalent to a user-defined type. The difference is that new fields can be added to either value part (identity class) or object part (entity class). For example, if each color has a unique name which has to be shared then we add the corresponding field to the entity class:

```
CONCEPT Color
   IDENTITY
      INT red, green, blue
   ENTITY
      CHAR(64) name
```

It is already a mix of three primitive values in the identity class and one object field. Now suppose that colors may have some other property which is supposed to be stored in a separate relation:

```
CONCEPT ColorDescr
   IDENTITY
      INT code
   ENTITY
      CHAR(2) lang
      CHAR(64) name
```

Now the color concept simply declares a field with this concept as a type:

```
CONCEPT Color
   IDENTITY
      INT red, green, blue
   ENTITY
      ColorDescr descr
```

Importantly, only one concept name is used as a type of variables in the model independent of its division on identity and entity parts.

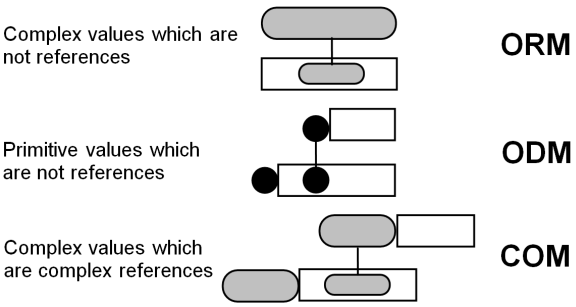


Figure 2. Value vs. object types in different approaches

All domains (independent of whether they are values, objects or both) are described using one system of types. If we use only identity classes then all elements in the model are values and it can be used for value modeling precisely as it is done in ORM. If all concepts have only entity part then we can model objects represented by OIDs or relations where rows are represented by surrogates. But in the general case, and it is one of the main advantages of COM, we can freely vary the division between values and objects. The differences between ORM, ODM and COM are shown in Fig. 2. In ORM, we can model only value domains which are used as attribute types. In ODM, we can model only object types. In COM, we can model both value types and object types.

3 Inclusion Instead of Inheritance

A place for everything and everything in its place — Victorian proverb

3.1 Modeling Hierarchies

In the previous section we asked the question *how* entities exist and assumed that the fact of their existence is manifested by means of identities. In this section, the main question is *where* objects exist. COM assumes that if something exists then there has to be some space,

environment, domain, container or context to which it belongs, that is, things are not able to exist outside of any space. Assuming so, the next question is what does it mean to exist *within* some space? The answer is that existence within a space means that the element is identified with respect to this space. Further assuming that the space is a normal element and any element is included only in one space then we get inclusion principle: *elements exist within a hierarchy where any element (excepting the root) is included in one parent with respect to which it is identified.*

Hierarchies are used in almost all branches of science and their descriptive role can hardly be overestimated. They exist almost everywhere in real life and they are especially useful in programming and data modeling. However, there exist many interpretations of hierarchies in terms of different relations:

- Containment hierarchies (inclusion relation)
- Re-use hierarchies (inheritance relation)
- Semantic hierarchies (specialization-generalization relation)

Containment hierarchies are used in the hierarchical data model [37] where many child elements are *included* in one parent and exist in its context. Hierarchies are also one of the corner stones of the object-oriented paradigm where they exist in the form of inheritance relation. Here the idea is that one base class can be extended by many more specific classes and in this way we can describe arbitrary entities from the problem domain by *reusing* more general descriptions. In semantic and conceptual models hierarchies are used to describe abstraction levels of the problem domain by defining more specific elements in terms of more general elements.

Yet, in spite of the highly general and natural character of the hierarchical view on data, it has not got a dominant position in data modeling. One reason for this state of affairs is asymmetry between the space of classes and the space of their instances [34]: classes exist within a hierarchy while their instances (objects) exist in flat space. Classical inheritance considers only class hierarchies and it is not possible to produce a hierarchy of their instances precisely what is needed in data

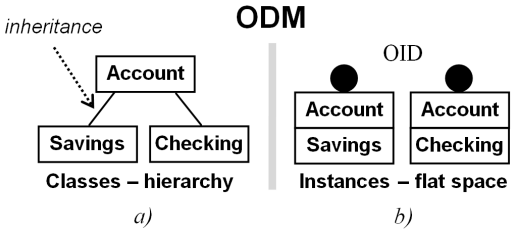


Figure 3. Asymmetry between classes and instances

modeling (and implemented in the hierarchical model). In other words, by using inheritance we cannot store objects in hierarchies like in the hierarchical model and it is a strong limitation because a database is modeled as a large *flat* space of objects.

For example, if two classes `Savings` and `Checking` extend one base class `Account` then we get a hierarchy where the parent class is shared among its child classes (Fig. 3a). Surprisingly, instances of these classes exist in a flat space so that each extension has its own parent and each instance is identified by one kind of OID (Fig. 3b). In other words, if two child classes have one shared parent class then their instances have no shared parent.

The conception of data reuse exists only at the level of classes but not their instances (code is reused in both cases). For data modeling purposes, it would be natural to create one main account object and then several savings and checking accounts within it. However, it is not possible using the traditional view on hierarchies, inheritance and semantic relationships. The goal of COM here is to unite the existing interpretations of hierarchies by introducing one relation for modeling containment (like in the hierarchical data model), inheritance (like in object data models), and generalization-specialization relation (like in semantic data models).

3.2 Inclusion for Modeling Hierarchies and Inheritance

COM revisits hierarchies by introducing a new relation, called *inclusion*. Inclusion relation assumes that child concepts are declared to be included in the parent concept. The most important property is that concept instances also exist within a hierarchy so that parents are shared parts of children. Note that the possibility to have many children within one parent is implied by the mechanism of identities. Each child is an instance of its concept and hence has some identity. This identity is always relative to the parent instance which has its own identity. For example, assume that concept `Savings` is included in concept `Account` (Fig. 4a):

```
CONCEPT Savings IN Account
   IDENTITY
      CHAR(2) subAccNo
   ENTITY
      DOUBLE balance
```

This concept declares its identity class having one dimension storing savings account number and its entity class having one dimension storing the current balance of the sub-account. It is important that sub-account number is a *relative* number which is meaningful only in the context of its main account. Hence (Fig. 4b) many savings accounts (instances of concept `Savings`) can be created in the context of one main account (an instance of concept `Account`). In particular, main account fields are shared among many savings accounts. In contrast, if `Savings` were a class inheriting from class `Account` then any new instance of `Savings` would get its own main account with all its fields.

An interesting observation about COM is that with the introduction of inclusion it essentially revives the hierarchical model of data but at the same time remains compatible with the object-oriented view. Inclusion hierarchy can be viewed as a nested container where each object has its own hierarchical domain-specific address which is analogous to a normal postal address. For example, a savings account might be identified by a reference consisting of two segments: $\langle'123456789'\rangle : \langle'02'\rangle$ where the first number is the main account and the second number

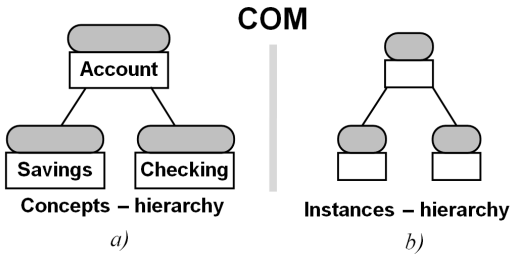


Figure 4. Symmetry between concepts and instances

identifies the relative savings account.

Note that these object identities are *virtual addresses* because they are defined in domain-specific terms which are not directly bound to the platform. Thus the represented entities can reside anywhere in the world. In particular, bank account objects can be persistently stored in a database and the account number is used to retrieve them. It is possible to specify how these virtual addresses are translated to physical addresses [30]. This can be used for implementing such mechanisms as replication, partitioning and distribution.

An important use of COM inclusion hierarchies consists in modeling types of values. If we define a concept without entity class then it will describe some value type. Using inclusion relation, this value can be extended by adding new dimensions and producing a more specific type. In this way we can build a hierarchy of value types. For example, we could define colors as elements identified by their unique name and having three entity properties:

```
CONCEPT Color IN ColorObject
   IDENTITY
      CHAR(16) name // For example 'red'
   ENTITY // Inherited from ColorObject
```

Of course, we could use conventional classes for this purpose like it is done in ORM but then we would need to have two kinds of classes for values and objects. The distinguishing feature of COM is that con-

cepts are intended for describing *both* values and objects as couples. In relational terms, COM allows for modeling *two* type hierarchies for domains and relations using only one construct (concept) and one relation (inclusion). In the general case, a domain in COM is a set of identity-entity couples. In this sense, it is a step in the direction of unifying object-oriented and relational models by uniting two orthogonal branches: domain modeling and relational modeling.

If concepts have empty identity classes then inclusion is a means for modeling relation types. For example, an existing concept `Persons` can be extended by introducing a new concept `Employees`:

```
CONCEPT Employees IN Persons
   IDENTITY // Empty
   ENTITY
      DOUBLE age
```

Due to the nature of inclusion relation, many employee records can belong to one person record which is obviously not what we need in this situation. In order to model classical extension, identity class of the `Employee` concept is left empty. In this case no more than one employee segment is possible because of the absent identity class. As a result, each person record will have one or zero extensions which means that a person belongs to either base `Person` concept or to the extended `Employee` concept.

4 Partial Order Instead of Graph

Ordnung muss sein — German phrase

4.1 Partial Order for Data Modeling

In the previous sections we described how objects exist and where they exist. In this section we answer the question what an object *means*, that is, what is its semantic content. When an element is created it gets some identity which determines its location in the hierarchical address space but says little about its relationships with other elements. To semantically characterize an element, it has to be connected with other

elements in the model. Almost all data models including ODM are implicitly or explicitly graph-based, i.e., they assume that a set of data elements is a graph. In this case semantics is encoded in relationships among elements. The distinguishing feature of COM is that it uses an approach where connectivity and semantics are represented by means of *partial order relation*, i.e., a database is defined as a partially ordered set of elements. As a result, any element participates in *two* orthogonal relations simultaneously: inclusion and partial order.

Strict partial order is a binary relation $<$ (less then) defined on elements of the set and satisfying the properties of irreflexivity and transitivity. If $a < b$ (a is less than b) then a is a lesser element and b is a greater element. In diagrams greater elements are positioned higher than lesser elements. Given two data elements, the main question in COM is whether one of them is less than the other. If we do this comparison for all elements then we get a concept-oriented database. Thus a concept-oriented database is a partially ordered set an example of which is shown in Fig. 5 where b, d and f are the greatest elements, and a and c are the least elements.

Formally, partial order in COM is represented by means of tuples by assuming that a tuple is less than any of its members: if $a = \langle \ldots, e, \ldots \rangle$ then $a < e$. For example, element a in Fig. 5 is less than b, d and e just because it is defined as $a = \langle b, d, e \rangle$. Thus if we have a number of tuples defined via each other then they induce partial order, and vice versa, if there is partial order then it can be represented by tuples representing elements and their members representing greater elements.

In object-oriented terms, this principle means that *references represent greater objects*. If one object references another object via one of its fields then the first object is less than the referenced object. For example, if a book element is an object storing a publisher, title and sales in its fields then these three constituents are its greater elements. Semantically, they are considered more general terms which define the meaning of the more specific book element. If we change a greater element then the meaning of all lesser elements which use it will also change.

To describe a partially ordered structure of elements at the level of

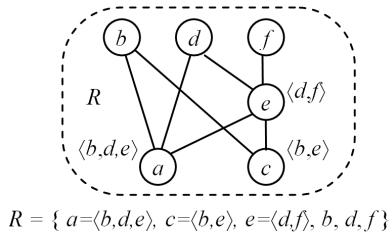


Figure 5. Concept-oriented database is a partially ordered set

concepts, COM uses the following principle: *dimension types represent greater concepts*. A set of concepts (with inclusion relation) partially ordered using this principle is referred to as a *concept-oriented schema* and a set of their instances is referred to as a *concept-oriented database*. Note that concepts and their instances participate simultaneously in *two* relations: inclusion and partial order.

For example, assume that concept `Book` has a dimension referencing its publisher of type `Publisher`:

```
CONCEPT Book // Book < Publisher
   IDENTITY
      CHAR(10) isbn
   ENTITY
      Publisher pub // Greater concept
      DOUBLE sales
```

According to this definition, `Book` is a lesser concept and `Publisher` is a greater concept: `Book < Publisher`. In object oriented approach, field types constrain possible elements that can be referenced via this field. In COM, we not only constrain possible referenced elements but also say that the referenced element is greater than this one and this property is then used for querying and reasoning about data.

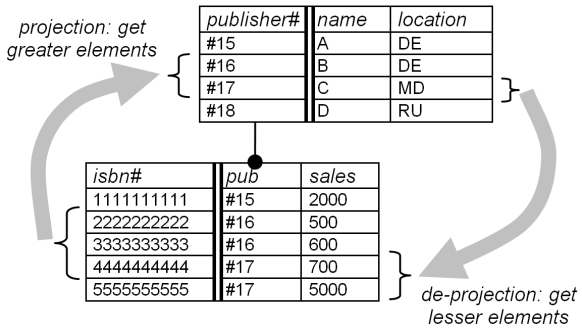


Figure 6. Projection and de-projection operations

4.2 Projection and De-Projection Operations

Data access operations in COM rely on the partially ordered structure of concepts and their instances. If we move up in the direction of some greater concept then this operation is called *projection* and denoted by right arrow `'->'`. If we move down in the direction of a lesser concept then this operation is called *de-projection* and denoted by left arrow `'<-'`. Given a set of elements, projection returns all greater elements and de-projection returns all lesser elements along the specified dimension. A sequence of projections and de-projections is called a *logical access path*. The main difference from graph-based models is that these operations are intended for changing the level of detail by moving only up and down in a zig-zag manner along the structure of dimensions. This makes COM similar to multidimensional models [22].

For example, assume that any book (`Books` collection) has one publisher (`Publishers` collection). Publishers of all books with low sales can be found by projecting the selected `Books` to the greater `Publishers` collection (Fig. 6):

```
(Books | sales < 1000) -> pub -> (Publishers)
```

All books of the selected publishers can be found using de-projection:

```
(Publishers | name == 'C') <- pub <- (Books)
```

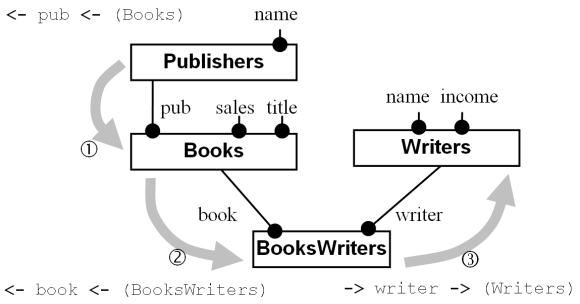


Figure 7. Logical navigation and inference in COM

Note that projection is a *set* of elements from the target domain without repetitions. For example, we could get all locations of the publishers as follows:

```
(Publishers) -> location
```

Here we do not write the primitive target domain because it can be restored from the schema and therefore is optional. The result of this query for the example in Fig. 6 consists of three values 'DE', 'MD' and 'RU' even though the value 'DE' occurs two times.

This approach is very convenient for retrieving related elements in complex schemas. Let us consider a schema in Fig. 7 with many-to-many relationship between books and writers where each book belongs to one publisher. All writers of the selected publisher can be retrieved for three steps using the following query:

```
(Publishers | name = 'XYZ')
   <- pub <- (Books)                    (1)
   <- book <- (BooksWriters)            (2)
   -> writer -> (Writers)               (3)
```

Here the selected publisher is (1) de-projected down to the collection of books, then (2) further down to the `BooksWriters` collection, and (3) finally the constrained facts are projected up to the `Writers` collection.

It is very easy to write correlated queries using this approach. For

example, assume that we need to find all books with sales higher than the average sales in their respective publishers. The average sales figure for a publisher is computed as follows:

```
AVG( pub <- (Books).sales) )
```

Now we simply select books which have sales higher than this number:

```
(Books | sales > AVG(pub <- (Books).sales) )
```

For comparison, in SQL this query has a rather complicated and not very intuitive form:

```
SELECT isbn FROM Books b WHERE
   sales > (SELECT AVG(sales) FROM Books WHERE
      pub = b.pub )
```

One advantage of our approach is that it does not use joins and therefore queries in COM are much simpler than in SQL. In COM, it is possible to use the conventional dotted notation however it does not remove the necessity to use joins. Therefore, dotted notation is used only when it is necessary to access individual elements. For set-oriented operations COM relies on projection and de-projection. An advantage is that these operations hide the underlying structure of identities and it is not necessary to change all queries if some identity has changed.

4.3 Reasoning about Data

An important consequence of having partial order with projection and de-projection operations is the ability to reason about data by *automatically* deriving conclusions from initial constraints. It is not necessary to specify an exact access path because the system is able to propagate initial constraints automatically. In traditional semantic and deductive models this can be done using inference rules which are treated differently and managed separately from data. In COM, data itself is treated inherently as dependencies and data schema is used for constraint propagation. Inference procedure is essentially integral part of the model because it does not rely on inference rules but rather uses data for reasoning about data. In other words, once a model has been defined, it can be immediately used for inference.

In our example it is possible to derive writers belonging to the selected publisher without specifying *how* it has to be done. Such a query provides only (i) what we have in the form of source constraints and (ii) what we want to get by specifying a target collection. The source constraints are then propagated to the target automatically by the system. For example, such a query could be written as follows:

```
GIVEN (Publisher | name = 'XYZ')
GET (Writers)
```

The most important feature of this query is that it does not specify *how* the result has to be obtained and therefore this query can be answered by only using some semantic data model and its ability to infer the result. Most semantic models are based on formal logic where the result is derived using inference rules. COM proposes an alternative approach to inference which relies on partially ordered structure of the database [29]. This procedure consists of the following two steps:

Down. Source constraints X are propagated down to the bottom collection Z using de-projection: $X \leftarrow\star Z$

Up. The constrained bottom collection Z is propagated up to the target collection Y using projection: $Z \star\rightarrow Y$

Here operators '$\leftarrow\star$' and '$\star\rightarrow$' (with star symbol) denote de-projection and projection along *all* dimension paths. Using this two-step inference procedure we can get all authors of one publisher using the following semantic query:

```
(Publishers | name = 'XYZ')
   <-* (BooksWriters) *-> (Writers)
```

Note that this query does not involve any dimension name. It says only that the inference has to be carried out using `BooksWriters` as a bottom collection. Thus the chosen `Publishers` are de-projected down to `BooksWriters` and then up to `Writers` which are returned as a result collection. Yet, this query can be further simplified if we unite two steps into one inference operator denoted `'<-*->'`. This operator works with only source constraints and a target:

```
(Publishers | name = 'XYZ') <-*-> (Writers)
```

To illustrate how rather complex queries can be written in a very concise form let us assume that we want to consider only small publishers (with less than 10 books) and return only writers who have written more than 2 books:

```
(Publishers p | p <- pub <- (Books) < 10)
   <-*-> (Writers w | w <-*-> (Books) > 2)
```

Here we use a shortcut that comparison of a collection with a number implies COUNT aggregation function, that is, the condition `p<-pub<-(Books) < 10` is true if the publisher has less than 10 books, and the condition `w <-*-> (Books) > 2` is true if the writer has more than 2 books.

4.4 Multiple Propagation Paths

Generally, there are two approaches to constraint propagation:

Direct. Source constraints are imposed directly on the elements of the source set and then they are propagated through the model.

Inverse. Source constraints are expressed in terms of the target elements and then imposed on them so that target elements are selected by specifying properties they have to satisfy.

Projection and de-projection operations are an example of the direct approach while SQL is an example of the second approach. One limitation of the constraint propagation procedure via projection and de-projection operations is that we cannot use many source constraints. For example, assume that the task is to find all books belonging to the selected publishers and writers (Fig. 7). In this case we have two constraints: a set of publishers and a set of writers. One solution is to use the second (traditional) approach by simply expressing these two source constraints as properties of the books:

```
(Books p | p.pub.name = 'Springer' AND
   p <-*-> (Writers | name = 'Smith') > 0 )
```

This query involves two explicit conditions imposed directly on the retrieved elements. The first condition selects books depending on

their publisher and the second condition selects books depending on their author. Then these two conditions are combined using logical 'AND' operation.

To solve this problem using the direct approach source constraints can be independently propagated and then the result is built as their intersection (also denoted by 'AND'):

```
(Publishers | name='Springer') <- (Books) AND
(Writers | name = 'Smith') <-*-> (Books)
```

This query consists of two propagation paths leading to the same target collection. The first operation propagates publishers to books and the second operation propagates writers to books. Then these two sets are combined using set intersection operation.

Strictly speaking the above query also contains a portion with inverse constraints where the names of publishers and writers are specified. These fragments can be removed by rewriting this query as follows:

```
'Springer'<- name <- (Publishers)
   <- pub <- (Books) AND
'Smith'<- name <- (Writers)
   <- writer <- (BooksWriters) -> book -> (Books)
```

The most important property of this query is that it does not involve inverse constraints at all. Both of its access paths start from some value and lead to the same target collection. The values are taken from primitive domains (text strings in this example). In contrast, inverse queries specify constraints as properties of collection elements. Of course, such a query is too verbose and in practice these two approaches are combined. For us it is important to understand that COM supports both approaches.

4.5 Analytical Queries

In previous sections we described how data can be retrieved from the database. In analytical applications, it is necessary to have a possibility to compute new values using existing data. For this purpose, COQL

introduces CUBE operator (also denoted as FOREACH in other papers) which combines several source collections and returns their product:

```
CUBE(Collection1, Collection2, ...)
BODY {
   ...
}
RETURN ...
```

This query allows us to iterate over all combinations of elements in the source collections and to perform intermediate calculations in its BODY block. The structure of the result is specified in the RETURN statement.

For example (Fig. 8), assume that each book belongs to some genre and writers live in certain countries. The goal is to show how book sales are distributed among genres and countries. The difficulty is that a book may have many authors living in different countries and we want to distribute the book sales evenly among the authors. To solve this problem, a derived method is added to the BooksWriters concept which computes sales for one book and one author:

```
CONCEPT BooksWriters
   IDENTITY ...
   ENTITY
      DOUBLE sales() {
         RETURN book.sales / book <- (BooksWriters)
      }
```

Here sales can be considered a normal (read-only) dimension which is computed for each instance of this concept by dividing the book sales by the number of book authors.

To compute sales for each genre and country, we build a cube any cell of which is one genre and one country:

```
CUBE(Genres g, Countries c)
```

Each element of the result set returned by this query, called a *cell* in OLAP, is a pair of one genre and one country. Now we can compute sales for each cell in the query body as follows:

```
CUBE(Genres g, Countries c)
BODY {
```

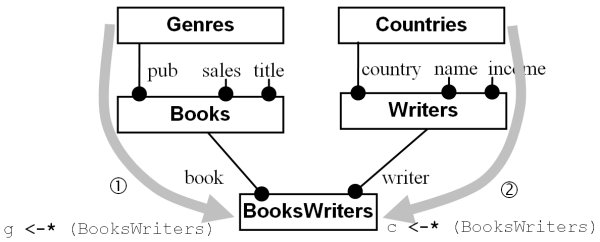


Figure 8. Multidimensional analysis in COM

```
    BW = g <-* (BooksWriters) AND
         c <-* (BooksWriters)
}
RETURN g.name, c.name, SUM(bw.sales)
```

Here `BW` is a cell and `BooksWriters` is a fact collection. A cell is computed as an intersection of all facts belonging to the current genre and all facts belonging to the current country. Then we simply return total sales for all facts in the cell where fact sales are computed in the derived method. Note that this approach does not use group-by operation which is replaced by de-projection.

5 Conclusion

The concept-oriented model is based on three general principles but intrinsically supports many patterns of thought used in other data models including set-based, hierarchical, multidimensional and semantic views on data. However, the largest overlap is with the object-based view and therefore COM can be characterized as taking its roots in object-orientation. In comparison with object-orientated approach, COM makes the following major contributions:

Concepts instead of classes. If class has only one constituent then concept combines two constituents in one construct: identity and entity. Data modeling is then reduced to describing identity-entity couples. This duality produces a nice yin-yang style of balance and sym-

metry between two orthogonal branches: identity modeling and entity modeling. In particular, this generalization allows us to model domain-specific identities instead of having only platform-specific ones. Concepts provide a basis for a new approach to type modeling. One its application is a unified mechanism for modeling relation types and domain types (which is one of the oldest problems in data modeling) by defining a domain as a set of identity-entity couples.

Inclusion instead of inheritance. Classical inheritance assumes that objects exist in one flat space where they are identified using one type of references. Inclusion generalizes this view by permitting objects to exist in a hierarchy where they are identified by hierarchical addresses. In this case both concepts and their instances exist within a hierarchy. This eliminates the asymmetry between classes defined as a hierarchy and their instances existing in flat space. Data modeling is then reduced to describing such hierarchical address space where objects are supposed to exist by focusing on identity modeling as opposed to traditional entity-centric approaches. This generalization turns objects into sets (of their children) and makes the whole approach inherently set-based rather than instance-based. Another important property of inclusion is that it retains all properties of classical inheritance and can be employed for reuse. Inheritance (IS-A) is revisited and considered a particular case of inclusion (IS-IN), that is, to put an object in a container means to inherit its properties.

Partial order instead of graph. Elements in COM are partially ordered where references represent greater elements and dimension types of concepts represent greater concepts. Data modeling is then reduced to ordering elements while other properties and mechanisms are derived from this relation. In particular, to be characterized by some property is equivalent to have another object as a greater element. Partial order also emphasizes the set-oriented nature of COM because elements are treated as sets of their lesser elements. Another important consequence of having partial order is that it allows us to implement an alternative approach to inference which is based on the data itself rather than on inference rules.

Due to this generality, COM decreases the existing mismatches be-

tween various kinds of models and methodologies:

Identity vs. entity and value vs. object. COM treats values and objects as two sides of one element by uniting identity modeling and entity modeling. In addition, COM provides an alternative approach for unifying domain and relational modeling.

Data modeling vs. programming. COM is integrated with a novel approach to programming, called concept-oriented programming (COP) [25, 30, 32] so that programming and data modeling are two branches of one methodology by decreasing the old and deeply rooted incongruity between these two branches of computer science [9, 8, 2]. In this context, COM can be defined as COP plus data semantics (implemented via partial order).

Transactional vs. analytical. COM unites transactional and analytical approaches to data modeling by narrowing the gap between operational systems and data warehouse (OLTP-OLAP impedance mismatch) by providing direct support for analytical operations which is currently a highly actual problem [23].

Logical vs. conceptual. COM is an inherently semantic data model which provides built-in mechanisms for reasoning about data by retaining conventional mechanisms for data access. In addition, COM essentially achieves the goals pursued by the nested relation model [14] and the universal relation models [15] but does it on the order-theoretic basis.

Instance-based vs. set-based. COM allows for both instance-level and set-level data access and manipulations.

Using the notions of concepts, inclusion and partial order, COM can describe a wide range of existing data modeling techniques and patterns. In particular, this approach does not use low level join and group-by operations. Taking into account its simplicity and generality, COM seems rather perspective direction for further research and development activities in the area of data modeling and object databases.

References

[1] S.Abiteboul, P.C.Kanellakis: Object identity as a query language primitive. *Journal of the ACM (JACM)*, 45(5): 798–842, 1998.

[2] M.Atkinson, P.Buneman: Types and Persistence in Database Programming Languages. *ACM Computing Surveys (CSUR)*, 19(2): 105–70, 1987.

[3] M.Atkinson, F.Bancilhon, D.DeWitt, K.Dittrich, D.Maier, S.Zdonik: The Object-Oriented Database System Manifesto. In Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases, 1990.

[4] F.Bancilhon: Object databases. *ACM Computing Surveys (CSUR)*, 28(1): 137–140, 1996.

[5] C.Chambers, D.Ungar, B.Chang, U.Hölzle: Parents are Shared Parts of Objects: Inheritance and Encapsulation in Self. *Lisp and Symbolic Computation*, 4(3): 207–222, 1991.

[6] E.Codd: A Relational Model for Large Shared Data Banks. *Communications of the ACM*, 13(6): 377–387, 1970.

[7] E.F.Codd: Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems (TODS)*, 4(4): 397–434, 1979.

[8] W.R.Cook, A.H.Ibrahim: Integrating Programming Languages and Databases: What is the Problem? *ODBMS.ORG*, Expert Article, 2006.

[9] G.P.Copeland, D.Maier: Making Smalltalk a Database System. *ACM SIGMOD Record*, 14(2): 316–325, 1984.

[10] K.R.Dittrich: Object-oriented database systems: the notions and the issues. *Proc. Intl. Workshop on Object-Oriented Database Systems*, 1986, 2–4.

[11] F.Eliassen, R.Karlsen: Interoperability and object identity. *ACM SIGMOD Record*, 20(4): 25–29, 1991.

[12] P.Hall, J.Owlett, S.Todd: Relations and Entities. *Modeling in DataBase Management Systems*, G.M.Nijssen (Ed.), North Holland, 1976, 201–220.

[13] R.Hull, R.King: Semantic database modeling: survey, applications, and research issues. *ACM Computing Surveys (CSUR)*, 19(3): 201–260, 1987.

[14] G.Jaeschke, H.J.Schek: Remarks On The Algebra Of Non First Normal Form Relations. *Proc. PODS'82*, 1982, 124–138.

[15] W.Kent, Consequences of assuming a universal relation. *ACM Transactions on Database Systems (TODS)*, 6(4): 539–556, 1981.

[16] W.Kent: A Rigorous Model of Object References, Identity and Existence. *Journal of Object-Oriented Programming*, 4(3): 28–38, 1991.

[17] S.N.Khoshafian, G.P.Copeland: Object identity. *Proc. OOPSLA'86*, ACM SIGPLAN Notices, 21(11): 406–416, 1986.

[18] G.M.Kuper, M.Y.Vardi: The Logical Data Model. *ACM Transactions on Database Systems*, 18(3): 379–413, 1993.

[19] H.Lieberman: Using prototypical objects to implement shared behavior in object-oriented systems. *Proc. OOPSLA'86*, ACM SIGPLAN Notices, 21(11): 214–223, 1986.

[20] J.Mylopoulos, Data Semantics Revisited: Databases and the Semantic Web, *DASFAA'04*, 2004.

[21] J.Peckham, F.Maryanski, Semantic data models. *ACM Computing Surveys (CSUR)*, 20(3): 153–189, 1988.

[22] T.B.Pedersen: Multidimensional Modeling. *Encyclopedia of Database Systems*. L.Liu, M.T.Özsu (Eds.) Springer, NY., 2009, 1777–1784.

[23] H.Plattner: A common database approach for oltp and olap using an in-memory column database. *Proc. SIGMOD'09*, 2009, 1–2.

[24] D.Raymond: *Partial order databases*. Ph.D. Thesis, University of Waterloo, Canada, 1996.

[25] A.Savinov: Concept as a Generalization of Class and Principles of the Concept-Oriented Programming. *Computer Science Journal of Moldova*, 13(3): 292–335, 2005.

[26] A.Savinov: Hierarchical Multidimensional Modeling in the Concept-Oriented Data Model. *Proc. 3rd Intl. Conference on Concept Lattices and Their Applications (CLA'05)*, 2005, 123–134.

[27] A.Savinov: Logical Navigation in the Concept-Oriented Data Model. *Journal of Conceptual Modeling*, Issue 36, 2005.

[28] A.Savinov: Grouping and Aggregation in the Concept-Oriented Data Model. *Proc. 21st Annual ACM Symposium on Applied Computing (SAC'06)*, 2006, 482–486.

[29] A.Savinov: Query by Constraint Propagation in the Concept-Oriented Data Model. *Computer Science Journal of Moldova*, 14(2): 219–238, 2006.

[30] A.Savinov: Concepts and Concept-Oriented Programming. *Journal of Object Technology*, 7(3): 91–106, 2008.

[31] A.Savinov: Concept-Oriented Model. *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends*, V.E.Ferraggine, J.H.Doorn, L.C.Rivero (Eds.), IGI Global, 2009, 171–180.

[32] A.Savinov: Concept-Oriented Programming. *Encyclopedia of Information Science and Technology*, 2nd Edition, M.Khosrow-Pour (Ed.), IGI Global, 2009, 672–680.

[33] A.Savinov: Concept-Oriented Query Language for Data Modeling and Analysis. *Advanced Database Query Systems: Techniques,*

Applications and Technologies, L.Yan, Z.Ma (Eds.), IGI Global, 2011, 85–101.

[34] L.A.Stein: Delegation Is Inheritance. *Proc. OOPSLA'87*, ACM SIGPLAN Notices, 22(12): 138–146, 1987.

[35] M.Stonebraker, L.Rowe, B.Lindsay, J.Gray, M.Carey, M.Brodie, P.Bernstein, D.Beech: Third generation database system manifesto. *ACM SIGMOD Rec.*, 19(3), 1990.

[36] M.Stonebraker: *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufmann, San Mateo, CA, 1995.

[37] D.C.Tsichritzis, F.H.Lochovsky: Hierarchical Data-Base Management: A Survey. *ACM Computing Surveys (CSUR)*, 8(1): 105–123, 1976.

[38] R.Wieringa, W.de Jonge: Object Identifiers, Keys, and Surrogates – Object Identifiers Revisited. *Theory and Practice of Object Systems*, 1(2): 101–114, 1995.

Alexandr Savinov,

SAP Research Dresden,
SAP AG
Chemnitzer Str. 48,
01187 Dresden, Germany
E–mail: *alexandr.savinov@sap.com*
Home page: *http://conceptoriented.org/savinov*

Received November 30, 2011

Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

A $\sqrt{\frac{N}{G}}$ Method for Generating Communication Sets

Rupali Bhardwaj, V.S. Dixit, Anil Kr. Upadhyay

Abstract

In the fully meshed network, where every node is connected directly to every other node, network traffic is very high because in the fully meshed network, number of communication links is $\frac{N\times(N-1)}{2}$ and communication cost is $2 \times N \times (N-1)$, where N is total number of nodes in the network. To minimize network traffic, we propose an algorithm for generation of communication sets that allows any two nodes to communicate by traversing at most two nodes regardless of the network size by dividing the nodes in the system into subgroups of size G where $G \geq 1$, which are then organized into quorum groups of size $k_1 = \left(\sqrt{\frac{N}{G}}\, approx.\right)$ in a method similar to that used in Maekawa's algorithm except that now quorum groups are constructed out of subgroups instead of nodes. The performance analysis of the proposed partitioning algorithm shows that it significantly reduces network traffic as well as total number of communication links required for a node to communicate with other nodes in the system.

Keywords: Quorum; Coterie; Communication sets; Network traffic

1 Introduction

Every node is connected directly to every other node in the completely connected network, so that number of communication links is very high, $\frac{N\times(N-1)}{2}$, where N is total number of nodes in the network. The number of hops used in a completely connected network is $N \times (N-1)$

because each node can reach every other node using one hop only. In the proposed partitioning algorithm each node can communicate with other nodes by either one or two hops regardless of the network size by dividing the nodes in the system into subgroups of size G where $G \geq 1$, which are then organized into quorum groups of size $k_1 = \left(\sqrt{\frac{N}{G}}\, approx.\right)$ in a method similar to that used in Maekawa's algorithm [1] except that now quorum groups are constructed out of subgroups instead of nodes. The idea presented in this paper is that the entire system is divided into number of subsets equal to the number of nodes. Each node in the system is assigned a subset of size k, $(k = \sqrt{N}\, approx.)$. After that, $\left(\frac{N}{G}\right)$ subgroups are formed with G subsets per subgroup. Now subgroups organized into $\left(\frac{N}{G}\right)$ quorum groups of size $k_1 = \left(\sqrt{\frac{N}{G}}\, approx.\right)$ in a method similar to that used in Maekawa's algorithm [1]. Now, each such Quorum group will be associated with G subgroups. The intersection between every pair of quorum groups is exactly one subgroup instead of a node. The performance of proposed partitioning algorithm will be evaluated using network traffic, communication links, communication cost and routing table size as a criterion for evaluation. The major contributions of the paper include: (i) a novel approach for generating communication sets proposed by considering the concept of Quorums (ii) simulation results show that proposed algorithm performed better than Hajj's [2] scheme with respect to reducing the network traffic and the total number of communication links. The remainder of the paper is organized as follows: Section 2 features existing research work in the field of generation of communication sets. Section 3 outlines the model on which our algorithm is based. The proposed partitioning algorithm is presented in section 4. Section 5 discusses about the performance of the proposed algorithm. Finally, section 6 concludes the paper.

2 Related Work

Maekawa's algorithm [1] is a distributed algorithm and total number of messages per mutual exclusion required $3\sqrt{N_k}$ messages: $\sqrt{N_k}$ messages to convey a request, $\sqrt{N_k}$ messages to obtain permissions, and $\sqrt{N_k}$ messages to release mutual exclusion. It is assumed that the nodes communicate only by passing messages instead of sharing of memory. Drawback of Maekawa's algorithm is that there is no procedure given for the construction of overlapped sets when $k-1$ is power of a prime number. Wassim EI-Hajj [2] presented a special network topology that is unique in terms of nodes interconnection, communication sets are designed by two techniques, when $p+1$ is the power of a prime number and when $p+1$ is not the power of a prime number. A distributed routing protocol is proposed by them after constructing the initial topology that allows any two sites to communicate with each other by traversing at most two nodes regardless of the network size. If k is power of a prime number then there exists a finite projective plane of order k. If either $k-1$ or $k-2$ is divisible by 4 and k is not a sum of two integral squares ($k \neq a^2 + b^2$), then finite projective plane of order k does not exist. $k \times (k+1) + 1$ lines are there in a finite projective plane of order k [3]. A method for creating a coterie with quorum size $k+1$, where k is a prime number is presented by K.T. Tseng, C.B. Yang [4]. In this paper [5], grid based quorums are constructed using paths that bear resemblance with billiard ball paths, through the resulting quorums of size $\sqrt{2N}$ as compared to $2\sqrt{N}$ of Maekawa's grid based method. Barbara [6] examined the relationship through pair wise non null intersections between weighted voting and sets of nodes. S. Rangarajan [7] proposed a distributed fault tolerant algorithm for the replica control problem that can be parameterized to achieve the desired balance between low message overhead and high data availability. M. Neilsen [8] introduced a new class of protocols within the unifying framework based on quorums which generalized all consensus protocols which used m rounds of message exchange. Load of a particular node is distributed over m identical nodes by partitioning the nodes into mutually overlapping subsets so that, through querying only a few

nodes, a node gets the partial system state information [9]. Advantage is that it significantly reduced the total number of messages required for a node to take scheduling decision.

3 The System Model

In our proposed algorithm, each node in the network allocates a communication set satisfying the following constraints:
A_1. $S_i \bigcap S_{j \neq \phi} \ \ \forall i, j \in 1, 2 \ldots N$.
A_2. S_i, $1 \leq i \leq N$ always contains i.
A_3. The size of $|S_i|$ is k for any i. That is,

$$|S_1| = |S_2| = |S_3| = \ldots\ldots = k$$

A_4. R_i is contained in k $S'_j s$ $\forall i, j \in 1, 2 \ldots N$.

Where set of subsets S_i is called coterie, R_i is referred to as requesting subset of S_i. According to Maekawa algortihm [1], for a network with N number of nodes, create N different sets of size k ($k = \sqrt{N}\, approx.$) such that N is represented as $N = k \times (k-1)+1$, where $k-1$ is a prime number. If N cannot be represented in this form, then some dummy sites have to be added for the construction to work.

Considering a system with 13 nodes, we see that the communication sets of the nodes 3, 7, 8 & 12 are as follows:
$S_3 = \{3, 7, 8, 12\}$, $S_7 = \{2, 7, 10, 13\}$,
$S_8 = \{1, 8, 9, 10\}$, $S_{12} = \{4, 5, 10, 12\}$.

Rule 1 states that there is at least one common node between the communication sets of any two nodes. Communication sets S_3 and S_7 have node n_2 in common as in Figure 1. Rule 2 states that each node should belong to its own communication set, i.e. node n_1 is part of communication set S_3. Rule 3 states that the size of each communication set is to be equal to k, i.e. size of S_3, S_7, S_8, S_{12} is 4 whenever $N = 13$. The fourth constraint states that each site should be contained in k other sets. Whenever communication sets are generated, each node can communicate with other nodes by either one or two hops regardless of the network size. So in case of S_3, node 3 will exchange its

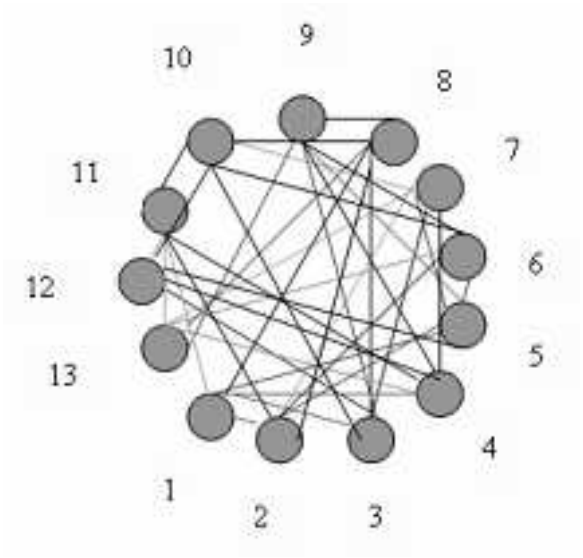


Figure 1. Network topology of size $N = 13$ nodes

state change messages only with node 7, 8 and 12. From these nodes it can also acquire the state information of nodes numbered 1, 2, 4, 5, 7, 8, 9, 10, 12 and 13 and can update its system state table also. So node 3 does not need to communicate explicitly with all the nodes except missing nodes 6, 11. So that total number of messages is equal to $2\times(k-1)+2\times$ *number of missing nodes.* Therefore, given a network of size N, our task is to generate a communication set for each node such that constraints A_1 through A_4 are satisfied.

4 The Proposed Partitioning Algorithm

Consider a system with N ($N \geq 2$) nodes, entire system is divided into number of subsets equal to the number of nodes. Each node in the system is assigned a communication set of size k, ($k = \sqrt{N}\, approx.$) according to algorithm 1 & 3. After that, we group these subsets into $\left(\frac{N}{G}\right)$ subgroups with G subsets per subgroup, where $G \geq 1$. We then

construct $\left(\frac{N}{G}\right)$ quorum groups such that each quorum group is made up of $k_1 = \left(\sqrt{\frac{N}{G}}\, approx.\right)$ subgroups; where each subgroup contains G subsets according to algorithm 1&3, except that now the quorum groups are constructed out of subgroups instead of nodes. Whenever Quorums are generated, each node can communicate with other nodes by either one or two hops regardless of the network size. Algorithm 2 calculates total number of missing nodes (worst case), through which each quorum communicated explicitly, because each quorum contained only partial system information.

4.1 Case 1: $N = k(k-1)+1$

Algorithm 1: Generating the communication sets where $k-1$ is a prime number
Result: Generate k groups of $(k-1)$ non-intersecting sets.
1. begin
2. Data K, count=2
3. Result: Generate a matrix B [K] [K-1]
4. for i = 0 to K do
5. for j = 0 to K-1 do
6. B[i] [j] = count++
7. end loop
8. input c = 1
9. Result: Calculate all $S[i]$ rows by performing operation on intersecting rows.
10. begin
11. for i = 0 to K do
12. S[c] [0] = 1
13. for j = 0 to K-1 do
14. S[c] [j+1] = B[i] [j]
15. end loop
16. c = B[i+1] [0]
17. end loop
18. end

19. input c = 2
20. Result: Calculate all $S[i]$ columns by performing operation on intersecting column.
21. begin
22. for j = 0 to K-1 do
23. S[c] [0] = 2
24. for k = 0 to K-1 do
25. S1[c] [k+1] = B [k+1] [j]
26. end loop
27. c = B[1] [j+1]
28. end loop
29. end
30. Result: Perform the operations of diagonal matrix
31. begin
32. set c = 0, s = 0, t2 = 2, z = 2
33. for k =1 to K-1 do
34. set t3 = 0, n = 0, r = 0
35. while (r < k-1) do
36. if (t3 < z-1) then
37. set t1 = B [t2] [++t3]
38. else
39. set t1 = B [0] [k]
40. set S1[t1] [0] = B[0] [k]
41. set c = 1+ n, j =1+ n, m1=1
42. for i =1 to k-1 do
43. S1[t1][m1] = B[i] [j-1]
44. m1++
45. c+ = k + s
46. P = c % (k-1)
47. if (P==0) then
48. set j = z
49. else
50. set j = P
51. end for loop
52. n = n + 1, r = r + 1

53. end while loop
54. end for loop
55. end
56. Result: Display the no. of S[i] matrix.
57. for i =1 to n do
58. for j = 0 to k do
59. print S1[i] [j]
60. end loop
61. end loop
62. end

Algorithm 2: Find the all missing nodes
1. begin
2. input l, k, c
3. for t1 = 1 to N do
4. initialize j = 0
5. for m1 = 0 to k do
6. l = S [t1] [m]
7. k = 0
8. while k <= K do
9. c = 0
10. for a = 0 to 1000 do
11. if SS[a]==S1[l] [K] then
12. c =1
13. break
14. end if
15. end for loop
16. if c = 0 then
17. SS[j] = S1 [l] [k]
18. j++
19. end if
20. end while loop
21. end step 4 for loop
22. for i = 0 to 1000 do
23. if S[i] != 0 then
24. print SS[i]

25. else
26. break
27. end if
28. end for loop
29. print t1
30. print N-i
31. end for loop of Step 2
32. end

4.2 Case 2: $N \neq k(k-1)+1$

When $N \neq k \times (k-1)+1$, we need to find the value of a number M such that $M = k \times (k-1)+1$, where $M > N$. First, we create a degenerate set of S_i's in a similar way as in algorithm 1 and eliminate $M - N$ S_i's from this coterie as well as these nodes must be replaced from each quorum such that M_1 replaced by N_1, M_2 replaced by N_2 etc.

Algorithm 3: Calculation of D and replacement of D sites
1. begin
2. k = $\sqrt{N}$ and let M = k (k-1) + 1
3. if M < N then
4. k = k + 1
5. D = M – N
6. Replace these D sites, D_1, $D_{2.....}$ from coterie as well as from quorums by N_1, $N_2 \ldots$ in such a way that each quorum size should be k in such a way that R_i is contained in more than k S_j's, for all $i, j \in 1, 2 \ldots N$.
7. If there is a duplication of node in S_i then insert a new node (starting from the first node) in such a way that R_i is contained in k S_j's, for all $i, j \in 1, 2 \ldots N$
8. end

An example of such grouping strategy is discussed here. We consider a system with 28 nodes; Figure 2 groups nodes with $\sqrt{N}$ nodes per subset. Figure 3 shows subgroups that contain four subsets ($G = 4$)

per subgroup and there are seven quorum groups $\left(\frac{N}{G}\right) = 7$ as shown in Figure 4. Each quorum group consists of $k_1 = 3$, $k_1 = \left(\sqrt{\frac{N}{G}}\, approx.\right)$ subgroups. The intersection between every pair of quorum groups is exactly one subgroup. Now construct communication sets for $N = 28$ by algorithm 1 and 3.

$S_1 = \{1, 2, 3, 4, 5, 6\}$
$S_2 = \{2, 7, 12, 17, 22, 27\}$
$S_3 = \{3, 11, 12, 18, 24, 27\}$
$S_4 = \{4, 8, 15, 17, 24, 28\}$
$S_5 = \{5, 8, 16, 19, 22, 27\}$
$S_6 = \{6, 11, 15, 19, 23, 27\}$
$S_7 = \{1, 7, 8, 9, 10, 11\}$
$S_8 = \{2, 8, 13, 18, 23, 28\}$
$S_9 = \{2, 9, 14, 19, 24, 26\}$
$S_{10} = \{2, 10, 15, 20, 25, 27\}$
$S_{11} = \{2, 11, 16, 21, 26, 28\}$
$S_{12} = \{1, 12, 13, 14, 15, 16\}$
$S_{13} = \{3, 7, 13, 19, 25, 28\}$
$S_{14} = \{3, 8, 14, 20, 26, 27\}$
$S_{15} = \{3, 9, 15, 21, 22, 28\}$
$S_{16} = \{3, 10, 16, 17, 23, 26\}$
$S_{17} = \{1, 17, 18, 19, 20, 21\}$
$S_{18} = \{4, 9, 16, 18, 25, 27\}$
$S_{19} = \{4, 10, 12, 19, 26, 28\}$
$S_{20} = \{4, 11, 13, 20, 22, 26\}$
$S_{21} = \{4, 7, 14, 21, 23, 27\}$
$S_{22} = \{1, 22, 23, 24, 25, 26\}$
$S_{23} = \{5, 9, 12, 20, 23, 28\}$
$S_{24} = \{5, 10, 13, 21, 24, 27\}$
$S_{25} = \{5, 11, 14, 17, 25, 28\}$
$S_{26} = \{5, 7, 15, 18, 26, 6\}$
$S_{27} = \{1, 27, 28, 26, 6, 8\}$
$S_{28} = \{6, 7, 16, 20, 24, 28\}$

Figure 2. Grouping nodes with $\sqrt{N}$ nodes per subset

$G_1 = (S_1, S_2, S_3, S_4)$
$G_2 = (S_5, S_6, S_7, S_8)$
$G_3 = (S_9, S_{10}, S_{11}, S_{12})$
$G_4 = (S_{13}, S_{14}, S_{15}, S_{16})$
$G_5 = (S_{17}, S_{18}, S_{19}, S_{20})$
$G_6 = (S_{21}, S_{22}, S_{23}, S_{24})$
$G_7 = (S_{25}, S_{26}, S_{27}, S_{28})$

Figure 3. Grouping subsets with G subsets per subgroup

$Q_1 = \{G_1, G_2, G_3\}$
$Q_4 = \{G_1, G_4, G_5\}$
$Q_6 = \{G_1, G_6, G_7\}$
$Q_2 = \{G_2, G_4, G_6\}$
$Q_5 = \{G_2, G_5, G_7\}$
$Q_7 = \{G_3, G_4, G_7\}$
$Q_3 = \{G_3, G_5, G_6\}$

Figure 4. Grouping subgroups with $\sqrt{\frac{N}{G}}$ subgroups per Quorum

5 Experimental Study

Performance of the proposed partitioning algorithm is evaluated using network traffic, communication links, communication cost and routing table size as a criterion for evaluation. According to simulation study, the proposed algorithm performs better than Hajj's [2] algorithm with respect to reducing the network traffic and the number of communication links. First, consider Hajjs [2] algorithm in some detail which is based on Maekawa's [1] algorithm.

5.1 Hajj's Algorithm

In Hajj's Algorithm [2], it is shown that how N sites in an ad-hoc network can be divided into communication sets with $\sqrt{N}$ nodes per communication set such that constraints A_1 through A_4 are satisfied. Communication between two nodes takes place either directly or through a third node, which will exist to connect them. An example is shown

in Figure 5 where there are 7 nodes and 7 communication sets with 3 nodes per communication set.
$S_1 = \{1, 2, 3\}$
$S_4 = \{1, 4, 5\}$
$S_6 = \{1, 6, 7\}$
$S_2 = \{2, 4, 6\}$
$S_5 = \{2, 5, 7\}$
$S_7 = \{3, 4, 7\}$
$S_3 = \{3, 5, 6\}$

Figure 5. Grouping nodes with $\sqrt{N}$ nodes per communication set

So in case of S_3, node 3 will exchange its state change messages only with node 5 and 6. From these nodes it can also acquire the state information of nodes numbered 1, 2, 5, 6 and 7 and can update its system state table also.

5.2 Simulation Result

5.2.1 Routing Table Size

Usually, each node stored a routing table of size $(N \times N)$ for making routing decisions. Now no routing table needs to be stored on any one of node, route is computed on demand.

5.2.2 Communication Links

The number of links in a completely connected network is $\frac{N \times (N-1)}{2}$; number of links required by our algorithm is $N \times \left(\sqrt{NG} - 1\right)$.

$$Gain = \left(\frac{N-1}{2 \times \left(\sqrt{NG} - 1\right)} \right) \tag{1}$$

5.2.3 Communication Cost

The number of hops used in a completely connected network is $N \times (N-1)$ because each node can reach every other node using one hop. In

the proposed algorithm each node can communicate with other nodes by either 1 or 2 hops. So, total number of hops is = $N\times$(no. of 1 hop + 2 $\times$(no. of 2 hops)) = $2 \times N \times \left(\sqrt{NG} - 1\right)^2$.

$$Loss = \left(\frac{N-1}{2 \times \left(\sqrt{NG} - 1\right)^2} \right) \tag{2}$$

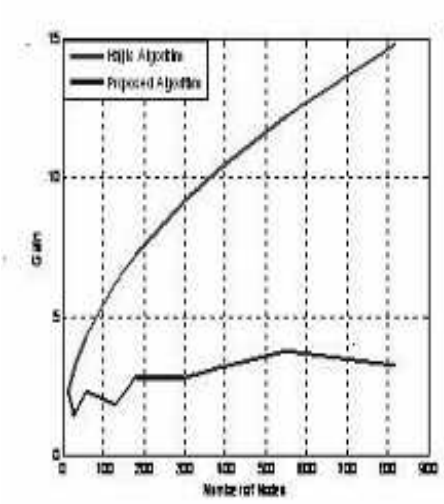


Figure 6. Gain

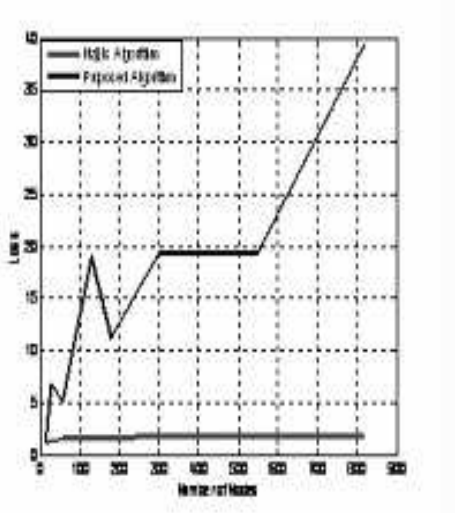


Figure 7. Loss

5.2.4 Network Traffic

Quorum contained only partial system information, so that some node information is missed through which each quorum communicated explicitly. If quorums explicitly communicate with these nodes, then the required number of messages will be $2 \times \{[(k-1) \times G]\} + 2\times$ *number of missing nodes* according to the proposed partitioning algorithm where, as in traditional systems, the number of required messages is $[2 \times (N-1)]$ and in Hajj's algorithm [2], the number of required messages is $[2 \times (k-1)+$ *number of missing nodes* $\times 2]$.

5.2.5 Analysis

It can be clearly seen from Figure 9 that network traffic is less in the proposed algorithm as compared to Hajj's algorithm [2], as N increases, traffic increases almost in a linear fashion. If communication sets are generated using the proposed algorithm, then the number of missing nodes is less as compared to Hajj's algorithm [2] (Figure 8). But in terms of gain (Figure 6) and loss (Figure 7), Hajji's algorithm [2] performs better than the proposed algorithm. As N increases, the gain increases almost in a linear fashion, while the loss is bounded by a constant in case of Hajj's [2] algorithm.

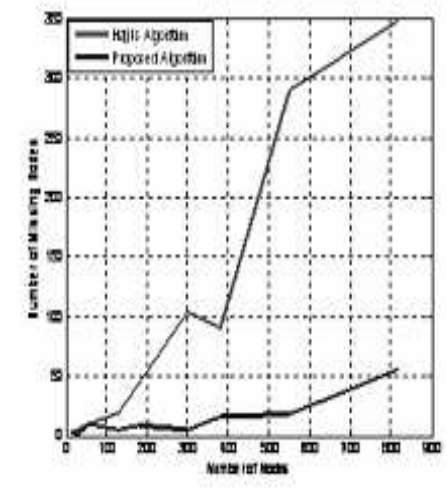


Figure 8. Missing Nodes Information

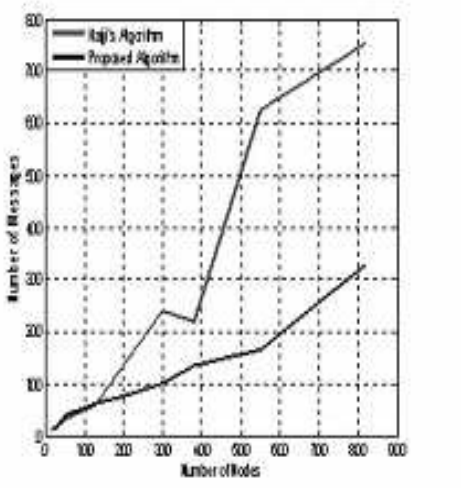


Figure 9. Network Traffic

6 Future work and conclusion

It is shown that from the perspective of the network traffic and partial system information, the proposed algorithm provides a significant performance over the traditional one. We illustrated by simulation that this protocol reduces both network traffic and number of messages communicated explicitly with missing nodes. Now, we will propose an algorithm for load balancing problem in P2P system using concept of

this paper.

References

[1] M. Maekawa. *A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems*, ACM Trans. Computer Systems,Vol. 3, pp.145–159, May 1985.

[2] Wassim El-Hajj, Hazem Hajj, Zouheir Trabelsi. *On Fault Tolerant Ad Hoc Network Design*, IWCMC'09, June 21-24, 2009,

[3] A. A. Albert, S.R. *An Introduction to Finite projective Planes*, New York: Holt, Rinehart, Winston, 1968.

[4] K.T. Tseng, C.B. Yang. *A $\sqrt{N}$ method for generating coteries*, M. Tech. thesis paper.

[5] Divyakant Agrawal, Omer Egecioglu, Amr El Abbadi. *Billiard quorums on the grid*, Information Processing Letters 64 (1997) 9–16, Elsevier Science.

[6] Hector Garcia-Molina, Daniel Barbara. *How to Assign Votes in a Distributed System*, J. ACM, Vol. 32, No. 4, pp. 841–860, 1985.

[7] Sampath Rangarajan, Sanjeev Setia, and Satish Tripathi. *A Fault-Tolerant Algorithm for Replicated Data Management*, IEEE Transactions on parallel and distributed systems, VOL. 6, NO. 12, December 1995.

[8] Mitchell L. Neilsen, Masaaki Mizuno. *Decentralized Consensus Protocols*, supported by National science Foundation under Grant CCR-8822378.

[9] Md. Abdur Razzaque1 and Choong Seon Hong. *Dynamic Load Balancing in Distributed System: An Efficient Approach*, funded by the "Korean Government (MOEHRD)" (KRF-2006-521-D00394), 2006.

A $\sqrt{\frac{N}{G}}$ Method for Generating Communication Sets

Rupali Bhardwaj, V.S.Dixit, Anil Upadhyay

Received January 27, 2011
Revised May 10, 2011

Rupali Bhardwaj
Institution: Krishna Institute of Engineering and Technology,
Mahamaya Technical University, Noida, India
Address: Department of MCA, KIET, Ghaziabad, India
E–mail: *rupalibhardwaj*09@*gmail.com*

Dr. V.S.Dixit
Institution: Atmaram Sanatan Dharmshala, Delhi University,Delhi,India
Address: Department of CS, ARSD College, Delhi University, Delhi, India
E–mail: *veersaindixit*@*rediffmail.com*

Anil Upadhyay
Institution: Mata Rajkaur Institute of Engineering and Technology,
Mahrishi Dayanand University, Rohtak, Haryana, India
Address: Department of Applied Science, MRKIET, Rewari, India
E–mail: *anilupadhyay*2005@*gmail.com*

Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

Coherent Route Cache In Dynamic Source Routing For Ad Hoc Networks

Sofiane Boukli Hacene, Ahmed Lehireche

Abstract

Ad hoc network is a set of nodes that are able to move and can be connected in an arbitrary manner. Each node acts as a router and communicates using a multi-hop wireless links. Nodes within ad hoc networks need efficient dynamic routing protocols to facilitate communication. An Efficient routing protocol can provide significant benefits to mobile ad hoc networks, in terms of both performance and reliability. Several routing protocols exist allowing and facilitating communication between mobile nodes. One of the promising routing protocols is DSR (Dynamic Source Routing). This protocol presents some problems. The major problem in DSR is that the route cache contains some inconsistence routing information; this is due to node mobility. This problem generates longer delays for data packets. In order to reduce the delays we propose a technique based on cleaning route caches for nodes within an active route. Our approach has been implemented and tested in the well known network simulator GLOMOSIM and the simulation results show that protocol performance have been enhanced.

Keywords: Ad Hoc Networks, Mobile Networking, Minimizing Delay, Stale routes problem, DSR.

1 Introduction

Each node in a mobile ad hoc network (MANET) is a router. Communication between nodes requires a multihop wireless path from a source to a destination, so nodes must cooperate in routing operation.

All nodes are mobile and can be connected dynamically in an arbitrary manner to form a network [1]. A challenge in the design of ad hoc networks is the development of dynamic routing protocols that can efficiently find routes between two nodes. The key task of routing protocols is to deliver packets from the source node to the given destination [2]. The existing routing protocols are, traditionally, divided into two classes, depending on when a node acquires a route to a destination. Reactive protocols invoke a route discovery procedure on demand only. Thus, when a route is needed, some sort of flooding-based global search procedure is employed. One of the promising reactive routing protocols is DSR. In general, routing protocol presents some problems, and one of the major problems in DSR is longer data packets delays caused by the search process in the cache. In this paper, we propose a technique to solve this problem.

The remainder of the paper is organized as follows. First, we give an overview on DSR and its operations. This is followed by focusing on stale routes problem in section 3. In section 4, a presentation of different works that tries to solve this problem is given. Next, we concentrate on the proposed technique to solve the problem of delay caused by stale routes in cache in section 5. In section 6, we present the performance evolution of the proposed approaches, and finally, we conclude in section 7.

2 Dynamic Source Routing Protocol (DSR)

Dynamic Source Routing [3, 4] is based on source routing, where the source node specifies the whole path to destination node in the packet header. When a source node needs to communicate with a destination node, it first searches in its route cache for a route to the destination, if a route is found, the source node uses it, otherwise the source node initiates a route discovery mechanism to discover a route. In a route discovery mechanism, the source node floods a route request message (RREQ) to neighboring nodes. The message contains the source address, the destination address, the request id, and a list containing the complete path to destination. When a node receives this request, it

proceeds as following:

- If the node has seen this same request before, it ignored the request.
- If the receiving node is the destination itself or a node having a route to the destination in its cache, it returns a route reply message (RREP), which contains: the source address, the destination address, and the route record in the route request message. The route reply message is sent back to the source node by following the same route record in the route request message in reverse order.
- Otherwise, it appends its own address to the route record, and rebroadcast the route request message to its neighboring nodes.

When the source node receives the route reply message, it starts sending data packets to the destination. When a route failure happens, the node upstream the broken link sends back to the effective source a route error message (RERR). Nodes receiving RERR message remove broken link from its routes cache. The source node initiates a route discovery if it receives RERR message, it still needs a route to the destination and no alternate route in its cache.

3 The Stale Routes Problem in the DSR Protocol

The DSR has the advantage of learning routes by scanning for information in packets that are received. A route from A to C through B means that A learns the route to C, but also that it will learn the route to B. The source route will also mean that B learns the route to A and C and that C learns the route to A and B. This form of active learning is very good and reduces overhead in the network, by this way each node in DSR can find alternative routes when link failure happens. This property will have a bad repercussion on route cache when node mobility is high. The route caches will contain in this case many stale

routes to destinations that may be used to reach a destination and this generates longer delay for data packets. Several previous studies deal with stale routes problem [5, 6].

4 Related Work

In this section we will present some ideas to enhance the DSR routing protocol.

Chen and Hou in [5] used a neighbor link-state information exchange mechanism. Once a connection has been established, the neighbor link-state information is exchanged among nodes along the route from the source to the destination. As the information of the neighbor lists is piggybacked in data packets, the nodes on the source route are able to learn the partial topology around the neighborhood of the connection. The simulation results show that with limited overhead incurred in neighbor list dissemination, the proposed protocol outperforms DSR with either path or link caches in terms of packet delivery ratio and route discovery overhead.

In [7], He et al propose an active packet technique to improve DSR. The mean idea is allowing a packet to visit each node twice. This packet is named "Active packet". The objective of the first visit is to obtain topology information of the network; and the objective of the second visit is to update route caches according to the obtained information. In the header active packet header contains a marker field to indicate if the packet is in the first or the second visit. The payload of the active packet is a connection matrix for the network topology. The active packet is generated periodically. Simulation results show that the method reduced the miss rates by up to 60% and routing packet numbers by up to 47%.

An enhancement to DSR by using a link breakage prediction algorithm was proposed in [8]. A mobile node uses signal power strength from the received packets to predict the link breakage time, and sends a warning to the source node only if the link is soon-to-be-broken. The source node can perform a pro-active route rebuild to avoid disconnection. Simulation results show that the method reduced the dropped

packets (by at least 20%). The tradeoff is an increase in the number of control messages by at most 33.5%.

5 The Proposed Technique

In order to minimize the delay which is experienced by data packets and reduce stale routes in caches, we add an expiration time for each route inserted in the cache. This idea is inspired from route management in the routing table of AODV [9, 10]. When learning new routes, a node must set an expiration time for each route inserted in the cache, and when this time expires, the route is removed from the route cache of the node. Each time a route is used the expiration time is set. The max value is fixed to 10 Seconds (represent approximately 1% of simulation time) empirically.

6 Performance Evaluation & Simulation Results

In order to evaluate the effectiveness of the proposed technique described above, we add it to the basic version 3 of DSR available in the GLOMOSIM simulator, and we compare it with the original version using performance metrics.

The simulation environment and the performance metrics used will be described in the next paragraph, the simulation results presentation and discussion is done later.

6.1 Simulation Environment

We have used the implementation of DSR version 3 included in the well known GlomoSim simulator. Our results are based on the simulation of 50 wireless nodes forming an ad hoc network moving about in a rectangular area of 1500 meters by 300 meters for 900 seconds of simulated time. The source-destination pairs are spread randomly over the simulation area, sending four data packets per second following a

CBR (constant bit rate) fashion. For our simulation 10-20-30 and 40 source-destination pairs are chosen. Traffic sessions are established randomly and stay active until the simulation ends. A random waypoint mobility model [3] is used. The movement scenario we used for each simulation is characterized by a pause time. Each node begins the simulation by selecting a random destination in the simulation area and moving to that destination at a speed distributed uniformly between 0 and 20 meters per second. It then remains stationary for pause time seconds. This scenario is repeated for the duration of the simulation. We carry out simulations with movement patterns generated for 10 different pause times starting by 0s varying by a step of 100s until 900s (the length of the simulation) is reached, which corresponds to limited motion. The physical radio characteristics of each mobile node's network interface, such as the antenna gain, transmission power, and receiver sensitivity, were chosen to approximate the Lucent WaveLAN direct sequence spread spectrum radio[11]. The performance metrics [12] used to evaluate performance are:

- **Average end-to-end delay of data packets**: This includes all possible delays caused by buffering during route discovery, queuing at the interface queue, retransmission delays at the MAC layer, and propagation and transfer times.
- **Communication overhead** is the total number of control packets, and including route request, route reply, and route error packets generated for each delivered data packet.
- **Number of broken links** is the number of invalid routes for sending data across it; the proposed technique reduces the use of the broken links.

6.2 Simulation Results and Discussions

We report the results of the simulation experiments for the original DSR protocol and for Optimized DSR (DSR Opt). In all figures below, Pause time varied between 0 seconds and 900 seconds. When pause

time is 0 seconds this denotes high mobility, while 900 seconds pause time means no mobility. Each scenario is repeated five times and the average values of the results are chosen.

6.2.1 Broken Links

In high mobility, the number of broken links is high (Figures 1, 2, 3 and 4). This is due to constant changement in the network topology and the incapability to find a valid alternative link. The number of sources also affects the number of broken links. When the number of sources increases, the number of broken links also increases because the need of more routes to destinations and the failure of one link can induce a breach of several communications. It can be noticed from those figures that DSR Opt results in substantially fewer link breaks, especially when pause times are small (high mobility). This is due to the expiration time mechanism added to DSR and consequently, the probability of using a stale route is minimized (The protocol tends to use fresher valid routes).

6.2.2 Average End-To-End Delay

In Figures 5, 6, 7 and 8 the results obtained for the end-to-end delay metric are presented. We observe that the end-to-end delay increases significantly when the number of sources increases, especially in high mobility because queues of nodes are almost full and nodes try to salvage many data packets. Minimizing stale routes contribute directly to minimizing end to end delay for data packets. When a broken link happens in DSR Opt, data packets experience a lower delay than in DSR because of the reduced number of cached route. The results show that DSR Opt outperforms DSR significantly when the number of sources is low and motion of nodes is high. This enhancement of DSR is suitable for Multimedia flows which cannot tolerate higher delays.

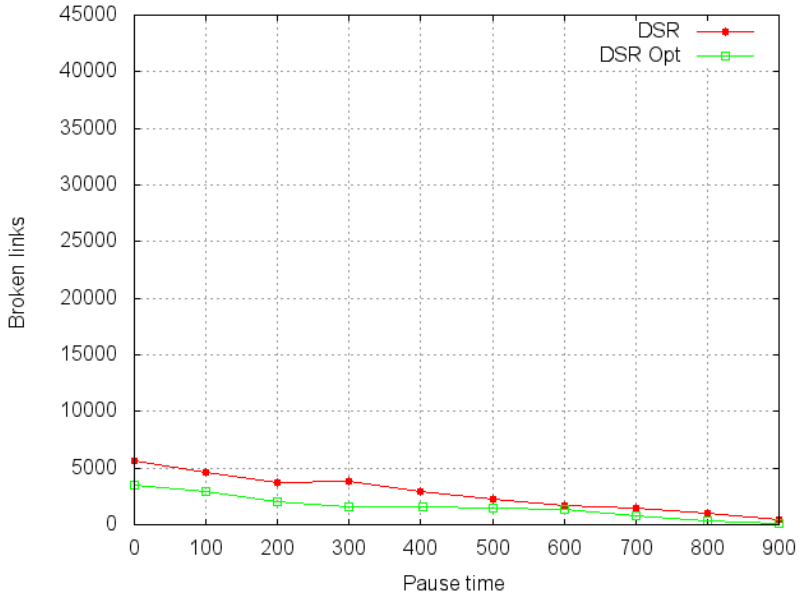


Figure 1. Number of Broken Links for 10 sources

6.2.3 Communication Overhead

Figures 9, 10, 11 and 12 show how mobility and number of sources affect the communication overhead. We notice that communication overhead is high when node mobility is high; this is due to the dynamic and constant change in network topology. It is also observed that the overhead is high when the number of sources is high. This results from the fact that many sources try to discover routes to destinations, which increase the number of control packets and so the communication overhead. The results show that DSR Opt results in substantially less overhead when the mobility is moderate (100s to 400s); this has a good impact on energy consumption because the number of control packets generated is low. Sometimes, DSR Opt generates higher overhead than DSR, this can be explained by the fact that when using the expiration time technique, some valid routes may be removed from the cache, which generates a new route discovery.

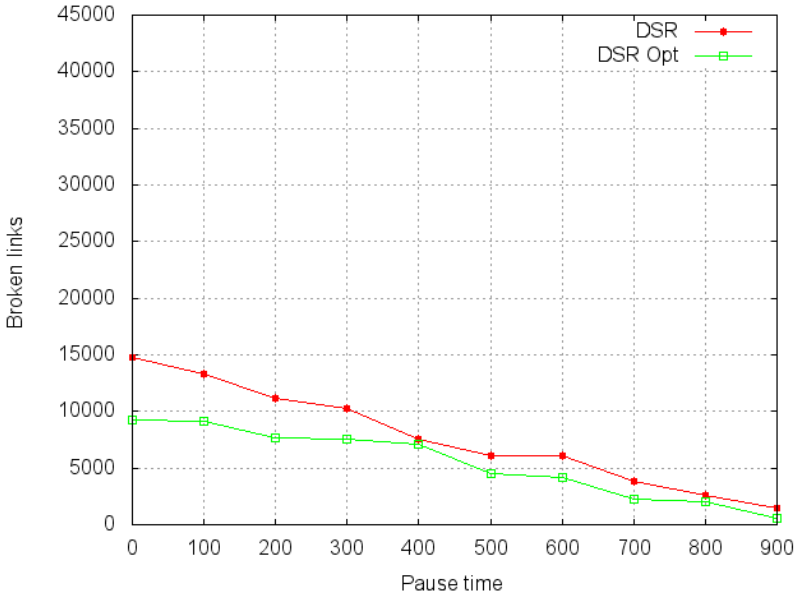


Figure 2. Number of Broken Links for 20 sources

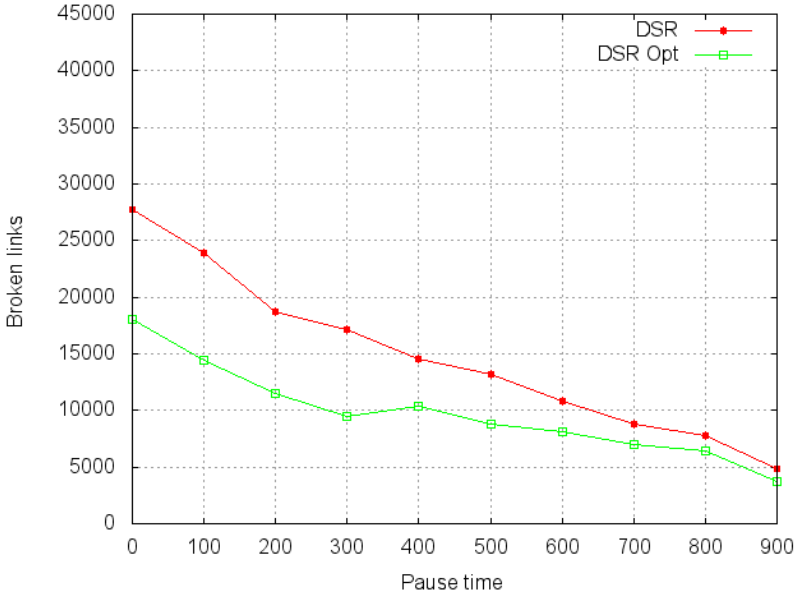


Figure 3. Number of Broken Links for 30 sources

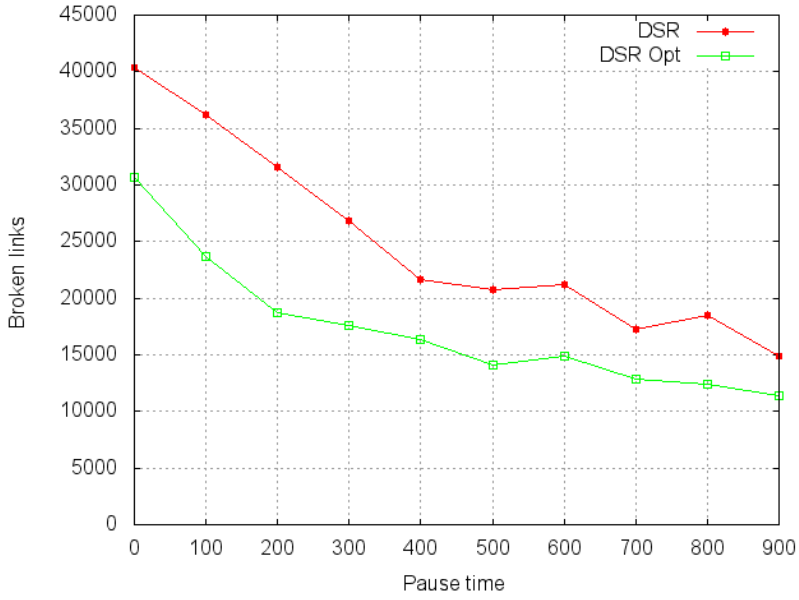


Figure 4. Number of Broken Links for 40 sources

7 Conclusion

In this paper, we have improved the promising DSR routing protocol for ad hoc networks. We have equipped DSR with expiration time technique for routes in route cache. This technique has been inspired from route management in the routing table of AODV routing protocol, in order to avoid the use of stale route in routing. The performance of the proposed technique was evaluated and compared with DSR using detailed simulations. Several common performance metrics were considered. The simulation results show that the proposed algorithm performs well; it can overall generate lower communication overhead, fewer broken links and lower Average end-to-end delay.

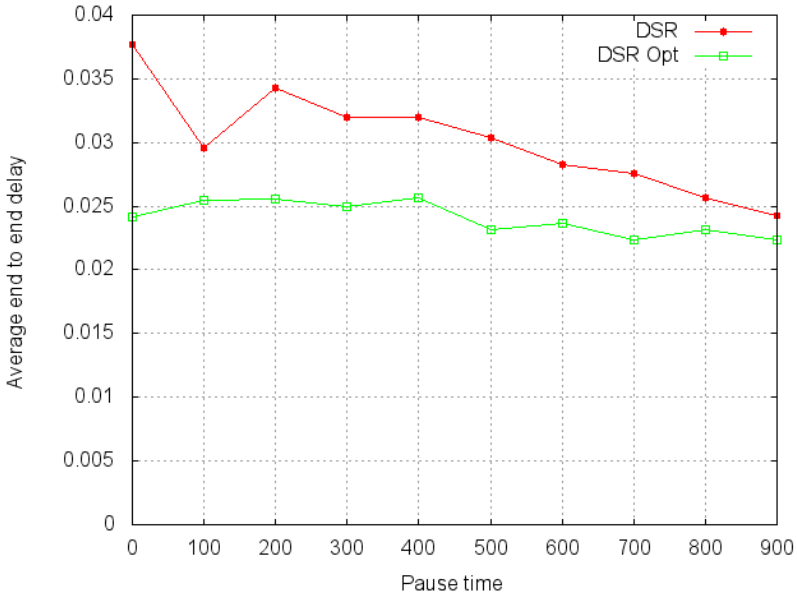


Figure 5. Average end to end delay for 10 sources

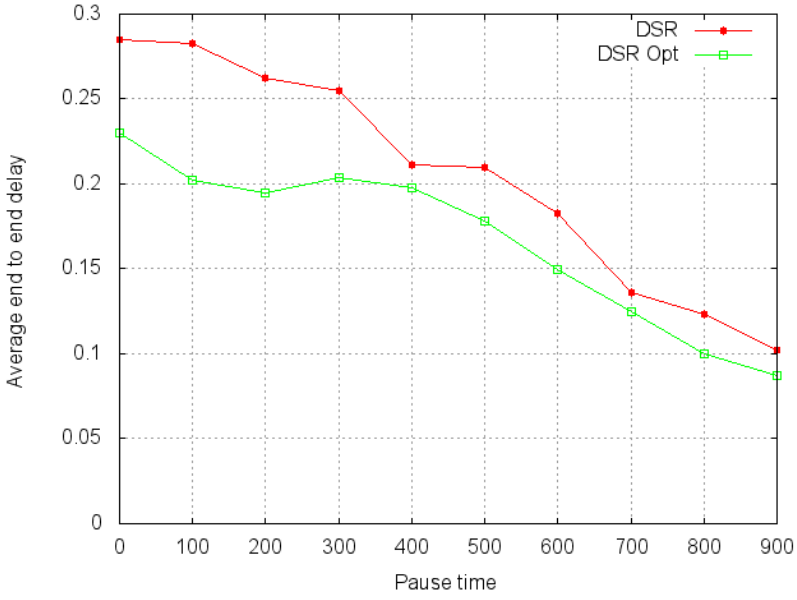


Figure 6. Average end to end delay for 20 sources

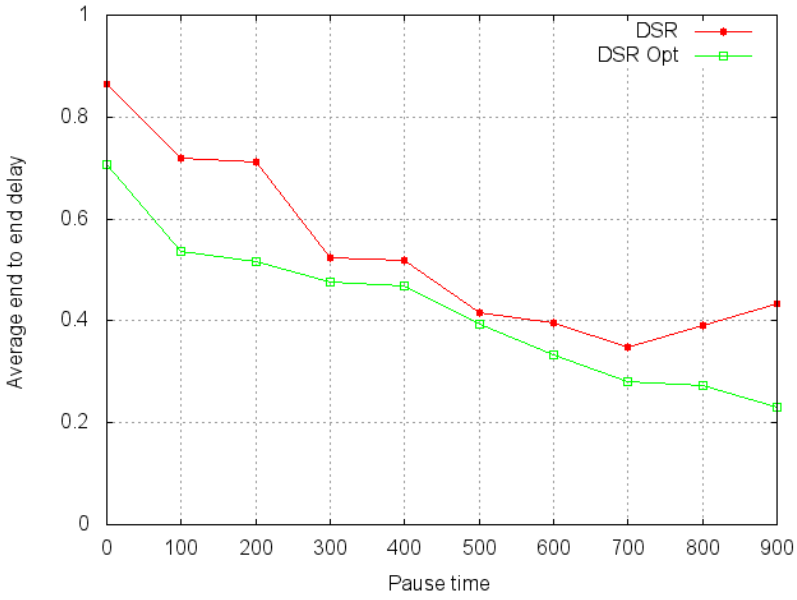


Figure 7. Average end to end delay for 30 sources

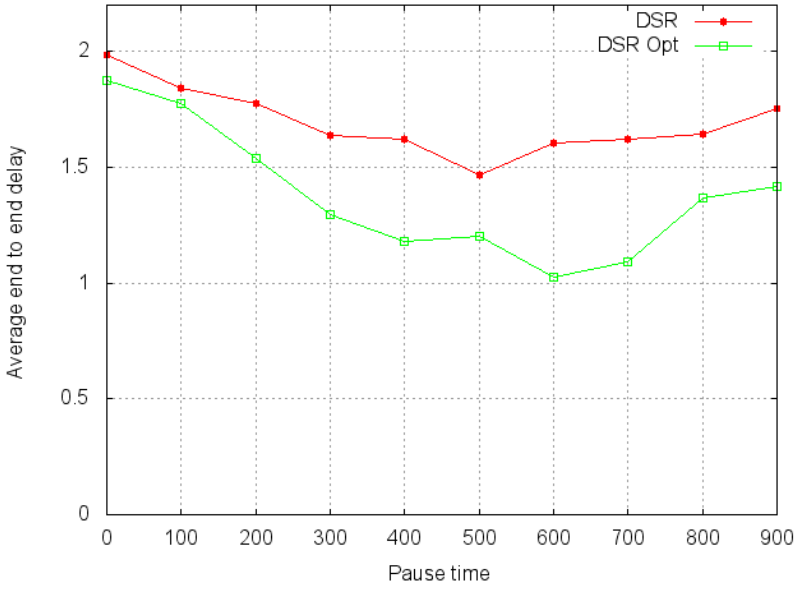


Figure 8. Average end to end delay for 40 sources

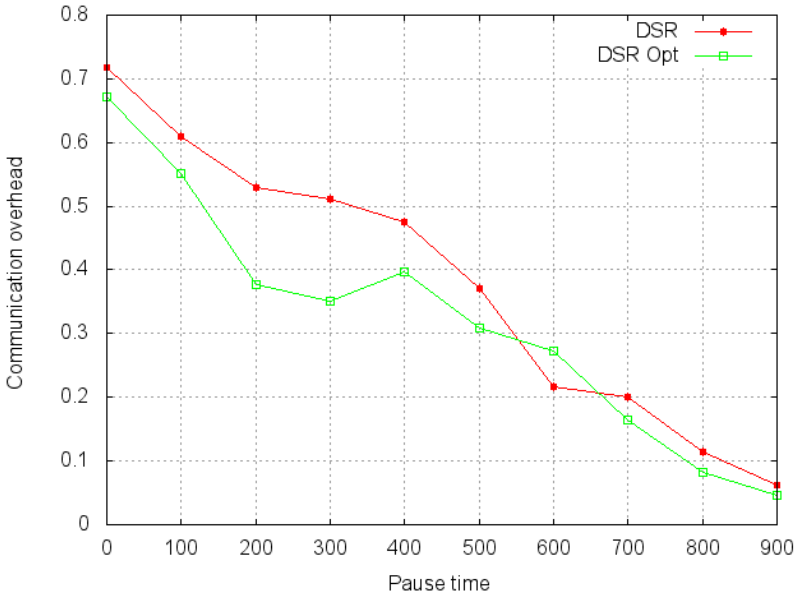


Figure 9. Communication overhead for 10 sources

References

[1] M. Amitava, B. Somprakash, and S. Debashis. *Location Management and Routing in Mobile Wireless Networks*. Wireless internet handbook: technologies, standards, and application, pp. 351–353, CRC Press, 2003.

[2] R. Ramjee and M. Marina. *Technology Trends in Wireless Communications*. Artech House, 2003.

[3] D. B. Johnson and D. A. Maltz. *Dynamic source routing in ad hoc wireless networks*. Mobile Computing Boston Academic, pp. 153–181, T. Imielinski and H. Korth (eds.), 1996.

[4] D. B. Johnson and D. A. Maltz. *Dynamic source routing in ad hoc wireless networks*. Internet Drafts, RFC Editor, 1996.

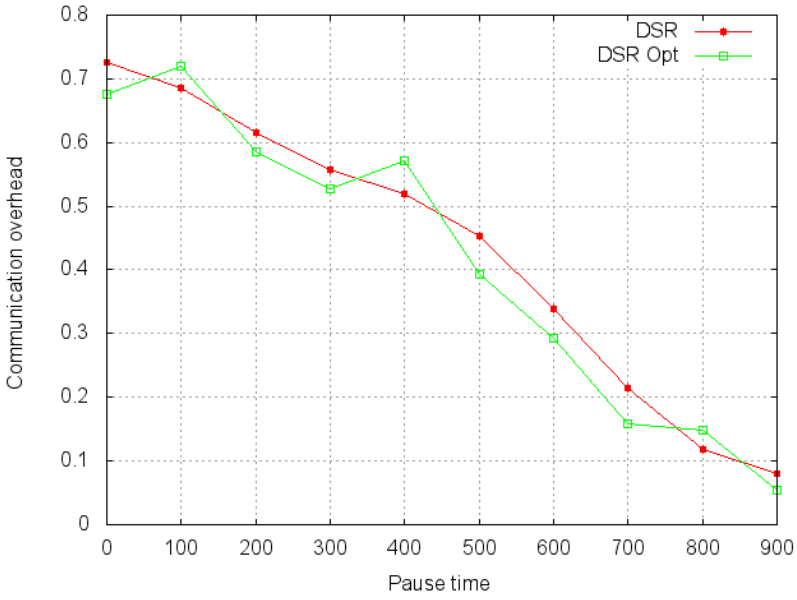


Figure 10. Communication overhead for 20 sources

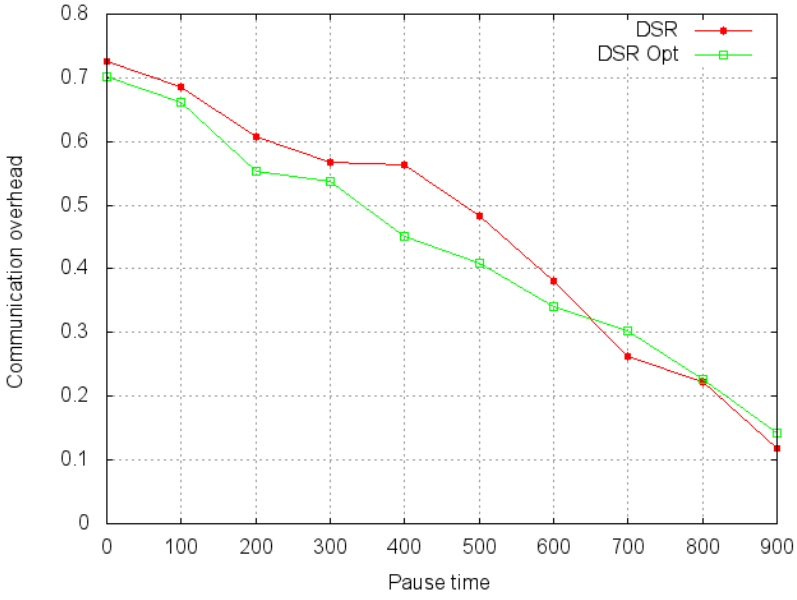


Figure 11. Communication overhead for 30 sources

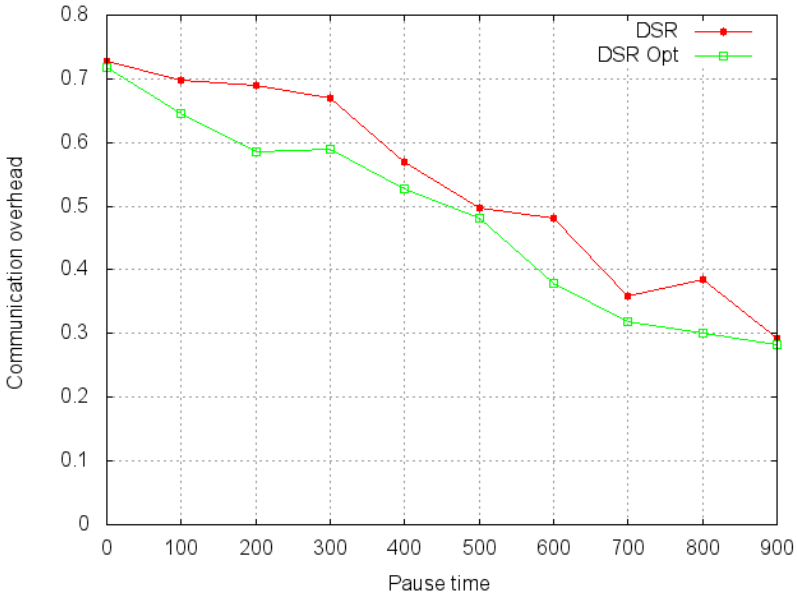


Figure 12. Communication overhead for 40 sources

[5] W. Chen and J. C. Hou. *Dynamic, Ad-Hoc Source Routing with Connection-Aware Like-State Exchange and Differentiation.* Proc. IEEE Globecom, vol. 1 (2002), pp. 188–194.

[6] A. Mathur. *Performance Analysis and Enhancement on Dynamic Source Routing for Mobile Ad Hoc Networks.* PHD Thesis in Computers Science. San Antonio: University of Texas at San Antonio, 2005.

[7] Y. He, C. S. Raghavendra, S. Berson, and B. Braden. *Active Packets Improve Dynamic Source Routing for Ad-Hoc Networks.* Next Generation Teletraffic and Wired/Wireless Advanced Networking, LNCS, vol. 47, No.12 (2007), pp. 367–378.

[8] L. Qin and T. Kunz. *Increasing packet delivery ratio in DSR by link prediction.* 36th Hawaii International Conference on Systems

Sciences (HICSS 03) Proceedings, Big Island, Hawaii, USA, January 2003, pp. 300–309.

[9] E. M. Belding-Royer and C. E. Perkins. *Evolution and future directions of the ad hoc on-demand distance-vector routing protocol*. Ad Hoc Networks, vol. 1, No.1 (2003), pp. 125–150.

[10] C. Perkins, E. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. Internet Drafts, RFC Editor, 2003.

[11] B. Tuch. *Development of Wavelan(R), an Ism Band Wireless Lan*. At&T Technical Journal, vol. 72, No.4 (1993), pp. 27–37.

[12] M. S. Corson and J. Macker. *Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. Internet Drafts, RFC 2501, IETF, Jan. 1999.

Sofiane Boukli Hacene, Ahmed Lehireche, Received July 07, 2011

Sofiane Boukli Hacene
Institution: Computer Science Department, Engineering Faculty, Djillali Liabes University, Sidi Bel Abbes, 22000, Algeria
Professional address: PO BOX 89, Computer Science Department, Engineering Faculty, Djillali Liabes University, Sidi Bel Abbes, 22000, Algeria
Personal address: 24, Rue A, cite serna, Sidi Bel Abbes, 22000, Algeria
Phone: 00213777881962
E–mail: *boukli@univ − sba.dz*, *boukli@gmail.com*

Ahmed Lehireche
Institution: Computer Science Department, Engineering Faculty, Djillali Liabes University, Sidi Bel Abbes, 22000, Algeria
Address: PO BOX 89, Computer Science Department, Engineering Faculty, Djillali Liabes University, Sidi Bel Abbes, 22000, Algeria
E–mail: *elhir@univ − sba.dz*

Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

Octagon Quadrangle Systems nesting 4-kite-designs having equi-indices

Luigia Berardi, Mario Gionfriddo, Rosaria Rota

Abstract

An *octagon quadrangle* is the graph consisting of an 8-cycle $(x_1, ..., x_8)$ with two additional chords: the edges $\{x_1, x_4\}$ and $\{x_5, x_8\}$. An *octagon quadrangle system* of order v and index λ $[OQS]$ is a pair $(X, \mathcal{B})$, where X is a finite set of v vertices and $\mathcal{B}$ is a collection of edge disjoint octagon quadrangles (called *blocks*) which partition the edge set of λK_v defined on X. A 4-*kite* is the graph having five vertices x_1, x_2, x_3, x_4, y and consisting of an 4-cycle $(x_1, x_2, ..., x_4)$ and an additional edge $\{x_1, y\}$. A 4-*kite design* of order n and index μ is a pair $\mathcal{K} = (Y, \mathcal{H})$, where Y is a finite set of n vertices and $\mathcal{H}$ is a collection of edge disjoint 4-*kite* which partition the edge set of μK_n defined on Y. An *Octagon Kite System* $[OKS]$ of order v and indices (λ, μ) is an $OQS(v)$ of index λ in which it is possible to divide every block in two 4-*kites* so that an 4-*kite design* of order v and index μ is defined.

In this paper we determine the spectrum for $OKS(v)$ nesting 4-kite-designs of *equi-indices* (2,3).

Lavoro eseguito nell'ambito del progetto PRIN 2008: *Disegni Combinatorici, Grafi e loro applicazioni.*

1 Introduction

Let $\lambda \cdot K_v$ be the complete multigraph defined on a vertex set X. Let G be a subgraph of $\lambda \cdot K_v$. A *G-decomposition* of $\lambda \cdot K_v$ is a

pair $\Sigma = (X, \mathcal{B})$, where $\mathcal{B}$ is a partition of the edge set of $\lambda \cdot K_v$ into subsets all of which yield subgraphs that are isomorphic to G. A G-decomposition is also called a *G-design* of order v and index λ; the classes of the partition $\mathcal{B}$ are said to be the *blocks* of Σ. Thus, $\mathcal{B}$ is a collection of graphs all isomorphic to G such that every pair of distinct elements of X is contained in λ blocks of Σ.

A 4-*kite* is a graph $G = C_4 + e$ (Fig.1), formed by a cycle $C_4 = (x, y_1, y_2, y_3)$, where the vertices are written in cyclic order, with an additional edge $\{x, z\}$. In what follows, we will denote such a graph by $[(y_1, y_2, y_3), (x), z]$. We will say that x is the *centre* of the kite, z the *terminal* point, y_1, y_3 the *lateral* points and y_2 the *median* point. A $(C_4 + e)$-design will also be called a 4-*kite-design*. It is known that a 4-kite-design of order v exists if and only if: $v \equiv 0$ or 1 mod 5, $v \geq 10$.

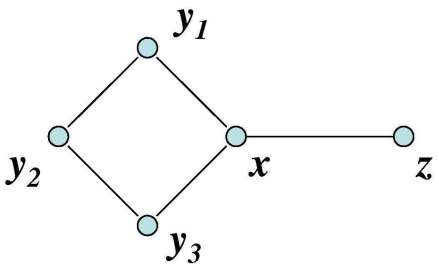


Figure 1. 4-kite

A λ-fold *m-cycle system* of order v is a pair $\Sigma = (X, \mathcal{C})$, where X is a finite set of v elements, called *vertices*, and C is a collection of edge disjoint m-cycles which partitions the edge set of λK_v, (the complete multigraph defined on X, where every pair of vertices is joined by λ edges). In this case, $|\mathcal{C}| = \lambda v(v-1)/2m$. The integer number λ is also called the *index* of the system. When $\lambda = 1$, we will simply say that Σ is an *m-cycle system*. The spectrum for λ-fold *m-cycle systems* for $\lambda \geq 2$ is still an open problem.

The graph given in Fig.2 is called an *octagon quadrangle* and will be denoted by $[(x_1), x_2, x_3, (x_4), (x_5), x_6, x_7, (x_8)]$. An octagon quad-

rangle system $[OQS]$ is a G-design, where G is an octagon quadrangle. $OQS(v)$s have been studied by the authors in [1][2][3][5].

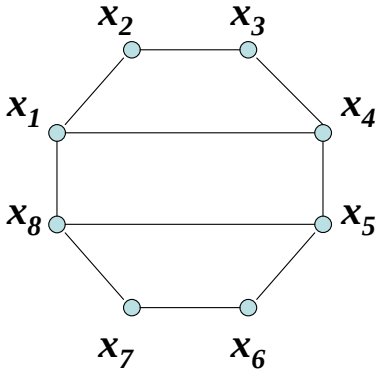


Figure 2. Octagon Quadrangle

Observe that, if we consider an *octagon quadrangle* $Q = [(x_1), x_2, x_3, (x_4), (x_5), x_6, x_7, (x_8)]$, it is possible to partition it into the two 4-kites $K_1(Q) = [(x_1, x_2, x_3), (x_4), x_5]$, $K_2(Q) = [(x_5, x_6, x_7), (x_8), x_1]$.

In this paper we study OQSs which can be partitioned into two $(C_4{+}e)$-designs (see Fig.3).

2 Definitions

Let $\Sigma = (X, \mathcal{B})$ be an OQS of order v and index λ. We say that Σ is 4-*kite nesting*, if for every octagon quadrangle $Q \in \mathcal{B}$ there exists at least a 4-kite $K(Q) \in \{K_1(Q), K_2(Q)\}$ such that the collection $\mathcal{K}$ of all these 4-kites $K(Q)$ form a 4-kite-design of order μ. This kite system is said to be *nested* in Σ. We will call it an *octagon 4-kite system* of order v and indices (λ, μ), briefly also OKS or $OKS_{\lambda,\mu}$, $OKS_{\lambda,\mu}(v)$.

If Ω is the family of all the 4-kites $\{K_1(Q), K_2(Q)\}$ contained in the octagon quadrangles $Q \in \mathcal{B}$, we observe that also the family $\mathcal{K}^c =$

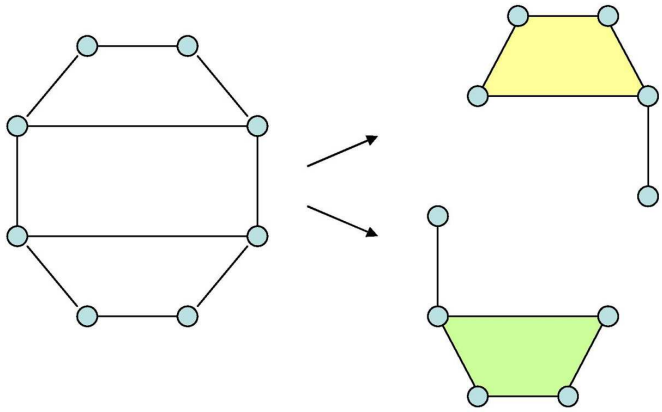


Figure 3. Decomposition of an OQ into two 4-kites

$= \Omega - \mathcal{K}$ forms a 4-kite-design of index $\mu' = \lambda - \mu$. If, for every octagon quadrangle $Q \in \mathcal{B}$, both families of 4-kites $\Omega_1 = \{K_1(Q) : Q \in \mathcal{B}\}$, $\Omega_2 = \{K_2(Q) : Q \in \mathcal{B}\}$ form a 4-kite design of index μ, we will say that the OQS is an *octagon bi-kite system.*

3 The new concept of G-Designs with equi-indices

In this section we give a new concept, which considers the possibility for a G-design to have more indices. We consider the case of *two* indices and order $v = 4h + 1$, because these will be considered in what follows, but the definition can be extended for the case with k indices, $k \geq 2$, and order every admissible v.

Definition of G-design with two equi-indices

Let G be a graph and let $v = 4h + 1$ an integer.
A G-design of equi-indices λ, μ is a pair $\Sigma = (X, \mathcal{B})$ where $X = Z_v$ and

$\mathcal{B}$ is a collection of graphs, all isomorphic to G, called blocks and defined in a subset of Z_v, such that for every pair of distinct element $x, y \in Z_v$:

1) *if the distance (difference) between x, y is equal to 1,2,...,h, then the pair x, y is contained in exactly λ blocks of Σ;*

2) *if the distance (difference) between x, y is equal to h+1,h+2,...,2h, then the pair x, y is contained in exactly μ blocks of Σ.*

This definition generalizes the well known concept of G-design and it can be done in many different ways and conditions. For them we keep the usual terminology.

Example 1

Let $v = 13$. In Z_{13} the set of all the possible differences is $\Delta = \{1, 2, 3, 4, 5, 6\}$. Partition Δ into the following two classes: $A = \{1, 2, 3\}$, $B = \{4, 5, 6\}$. It is possible to define a K_3-design (*Steiner Triple System*) of order $v = 13$ and *equi-indices* (λ, μ)=(1,2), as follows:

$$\forall\{x, y\} \subseteq Z_{13},\ \ x \neq y,\ \ |x - y| = 1, 2, 3 \Longrightarrow \lambda = 1,$$

$$\forall\{x, y\} \subseteq Z_{13},\ \ x \neq y,\ \ |x - y| = 4, 5, 6 \Longrightarrow \mu = 2,$$

where $\lambda = 1$ and $\mu = 2$ mean respectively that the pair x, y is contained in exactly one or two blocks of the system.

The blocks

$$\{i, i + 1, i + 5\}, \{i, i + 2, i + 7\}, \{i, i + 3, i + 7\},$$

for every $i \in Z_{13}$,

define such a system.

Example 2

Let $v = 9$. In Z_9 the set of all the possible differences is $\Delta = \{1, 2, 3, 4\}$. Partition Δ into the following two classes: $A = \{1, 2\}$, $B = \{3, 4\}$. It is possible to define a $(K_4 - e)$-design of order $v = 9$ and *equi-indices* (λ, μ)=(3,2), as follows:

$$\forall\{x, y\} \subseteq Z_9, x \neq y, |x - y| = 1, 2 \Longrightarrow \lambda = 3,$$
$$\forall\{x, y\} \subseteq Z_{13}, x \neq y, |x - y| = 3, 4 \Longrightarrow \mu = 2,$$

where $\lambda = 3$ and $\mu = 2$ mean respectively that the pair x, y is contained in exactly three or two blocks of the system.
If we indicate by $[x, y, (z, t)]$ a block of a $(K_4 - e)$-design, where (z, t) are the only two vertices non adjacent in $(K_4 - e)$, then the blocks

$$\{i, i + 2, (i + 1, i + 4)\}, \{i, i + 4, (i + 1, i + 6)\},$$

for every $i \in Z_9$,

define such a system.

4 Strongly Balanced 4-kite-Designs

It is known that a G-design Σ is said to be *balanced* if the *degree* of each vertex $x \in X$ is a constant: in other words, the number of blocks of Σ containing x is a constant.

In [4] the authors have introduced the following concept.
Let G be a graph and let $A_1, A_2, ..., A_h$ be the orbits of the automorphism group of G on its vertex-set. Let $\Sigma = (X, \mathcal{B})$ be a G-design.

We define the degree $d_{A_i(x)}$ of a vertex $x \in X$ as the number of blocks of Σ containing x as an element of A_i.

We say that:

$\Sigma = (X, \mathcal{B})$ *is a strongly balanced* G*-design if, for every* $i = 1, 2, ..., h$*, there exists a constant* C_i *such that*

$$d_{A_i}(x) = C_i,$$

for every $x \in X$*.*

It follows that:

Theorem 4.1. *A strongly balanced* G*-design is a balanced* G*-design.*

Proof Clearly, if $\Sigma = (X, \mathcal{B})$ is a *balanced* G-design, then for each vertex $x \in X$ the relation $d(x) = \sum_{i=1}^{h} d_{A_i}(x)$ holds. Hence, there exists $k \in N$ such that $d(x) = k$, for every $x \in X$. □

We say that a G-design is *simply balanced* if it is balanced, but not strongly balanced.

Theorem 4.2. *If* $\Sigma{=}(X, \mathcal{B})$ *in a balanced OQS of order* v *and index* λ*, then* Σ *is strongly balanced.*

Proof If G is an octagon quadrangle $[(x_1), x_2, x_3, (x_4), (x_5), x_6, x_7, (x_8)]$, the automorphism group of its vertices has the two orbits $A_1 = \{x_1, x_4, x_5, x_8\}$, $A_2 = \{x_2, x_3, x_6, x_7\}$. For every vertex $x \in X$, denote by C_x the number of blocks of Σ containing x as a central vertex, i.e. in a position of degree three, and by M_x the number of blocks of Σ containing x as a median vertex, i.e. in a position of degree two. Since Σ is balanced, it follows that:

$$C_x + M_x = d(x) = \frac{8 \cdot |\mathcal{B}|}{v} = 4 \cdot \lambda \cdot (v - 1);$$

$$4 \cdot C_x = \frac{8 \cdot |\mathcal{B}|}{v} = 4 \cdot \lambda \cdot (v - 1).$$

Hence:

$$C_x = M_x = 2 \cdot \lambda \cdot (v - 1),$$

and the statement is so proved. □

If Ω is a balanced 4-kite-design, it is possible that Ω is not strongly balanced. In Fig.4 there is a *simply balanced* 4-kite- design of order 11. We can see that all the vertices x have degree $d(x) = 5$, but the vertex 2 has *median-degree* equal to *three*, while the vertex 3 has *median-degree* equal to *zero*.

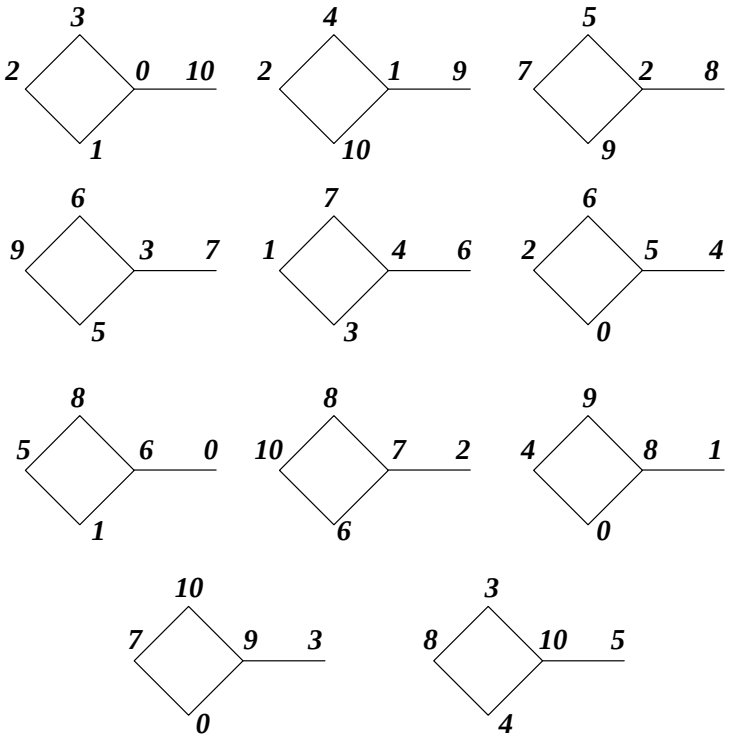


Figure 4. Simply balanced 4-kite-design of order 11

5 Necessary existence conditions

In this section we prove some necessary conditions for the existence of 4-kite-designs.

Theorem 5.1. *Let* $\Omega = (X, \mathcal{B})$ *be an* $OKS_{\lambda,\mu}(v)$. *Then*

i) $\lambda = 2 \cdot \mu$;

ii) (λ, μ)=*(2,1) or (4,2) or (6,3) or (8,4) implies* $v \equiv 0,1 \bmod 5$, $v \geq 8$;

iii) (λ, μ)=*(10,5) implies* $v \equiv 0,1 \bmod 2$, $v \geq 8$.

Proof Let $\Omega = (X, \mathcal{B})$ be and let $\Sigma = (X, \mathcal{K})$ the 4-kite system of index μ, nested in it. It follows that:

$$|\mathcal{B}| = |\mathcal{K}|,$$

$$|\mathcal{B}| = v(v-1) \cdot \lambda/20,$$

$$|\mathcal{K}| = v(v-1) \cdot \mu/10.$$

Hence: *i)* $\lambda = 2 \cdot \mu$. For *ii)* and *iii)*, it suffices to consider that:

$$|\mathcal{K}| = v(v-1) \cdot \mu/10. \qquad \square$$

Theorem 5.2. *Let* $\Sigma = (Z_v, \mathcal{B})$ *be a 4-kite-design of equi-indices* (λ, μ), *with* v *odd. Then*

i) $|\mathcal{B}|$=$(\lambda + \mu) \cdot v \cdot (v-1)/20 \in N$;

ii) (λ, μ)=$(2, 3)$ *implies* $v \equiv 1 \bmod 4$, $v \geq 5$.

Proof *i*) Let $\Sigma = (Z_v, \mathcal{B})$ be a 4-kite-design of equi-indices (λ, μ), with v odd. It follows:

$$|\mathcal{B}| = (\frac{\lambda}{2} \cdot \binom{v}{2} + \frac{\mu}{2} \cdot \binom{v}{2})/5 = \frac{\lambda + \mu}{10} \cdot \binom{v}{2},$$

which must be a positive integer. *ii*) Directly from *i*). □

The *ii*) of Theorem 5.2 will be used in the next section.

6 Main Existence Theorems

In this section we prove the conclusive Theorems of this paper. In what follows, if $B = [(a), b, c, (d), (\alpha), \beta, \gamma, (\delta)]$ in a block of a system Σ defined in Z_v, then the *translates* of B are all the blocks of type $B_j = [(a+j), b+j, c+j, (d+j), (\alpha+j), \beta+j, \gamma+j, (\delta+j)]$, for every $j \in Z_v$. B is called a base block of Σ.

Theorem 6.1. *There exists an OKS of order v and equi-indices $(2, 3)$, with v odd, if and only if* :

$$v \equiv 1 \ mod \ 4, \ \ v \geq 9.$$

Proof $\Rightarrow$ Let $\Sigma = (Z_v, \mathcal{B})$ be an OQS of order v and equi-indices $(2, 3)$, with v odd. Since every block contains eight vertices, from Theorem 5.2.*ii*), it follows

$$v \equiv 1 \ mod \ 4, \ \ v \geq 9.$$

$\Leftarrow$ Let $v = 4h + 1, \ \ h \in N, \ \ h \geq 2$.
Consider the following octagon quadrangles:

$$B_1 = [(0), h, 3h+1, (1), (2h+1), 3h, h+1, (2)],$$

$$B_2 = [(0), 1, 3h+1, (2), (2h+1), 3h-1, h+1, (3)],$$

$B_3 = [(0), 2, 3h+1, (3), (2h+1), 3h-2, h+1, (4)]$,

.....

$B_i = [(0), i-1, 3h+1, (i), (2h+1), 3h-(i-1), h+1, (i+1)]$,

.....

$B_{h-1} = [(0), h-2, 3h+1, (h-1), (2h+1), 3h-(h-2), h+1, (h)]$,

$B_h = [(0), h-1, 3h+1, (h), (2h+1), h+1, 3h+2, (1)]$.

Consider the system $\Sigma = (X, \mathcal{B})$, defined in $X = Z_v$, having $B_1, ..., B_i, ..., B_h$ as *base blocks*. This means that $B_1, B_2, ..., B_i, ..., B_h$ belong to $\mathcal{B}$ and also all the *translates*.

It is possible to verify that Σ is an OQS of order $v = 4h + 1$ and index $\lambda = 5$. Further, if we divide every block $Q = [(x_1), x_2, x_3, (x_4), (x_5), x_6, x_7, (x_8)]$, into the two 4-kites

$$K_1(Q) = [(x_1, x_2, x_3), (x_4), x_5],$$

$$K_2(Q) = [(x_5, x_6, x_7), (x_8), x_1],$$

we can verify that the collection of all the *upper* 4-kites form a 4-kite-design $\Sigma_1 = (Z_v, \mathcal{B}_1)$ of equi-indices $(\lambda = 2, \mu = 3)$, while the collection of all the *lower* 4-kites form a 4-kite-design $\Sigma_2 = (Z_v, \mathcal{B}_2)$ of equi-indices $(\lambda = 3, \mu = 2)$.

Observe that:

i) in Σ_1 all the pairs $x, y \in Z_v$ associated with the index $\lambda = 2$ have difference $|x - y|$ belonging to $A = \{1, 2, ..., h\}$, while the pairs associated with $\mu = 3$ have difference belonging to $B = \{h+1, h+2, ..., 2h\}$;

ii) in Σ_2 all the pairs $x, y \in Z_v$ associated with the index $\lambda = 3$ have difference $|x - y|$ belonging to A, while the pairs associated with $\mu = 2$ have difference belonging to B.

This proves that Σ is an OKS of order $v = 4h + 1$, $h \geq 2$, where the two 4-kite-designs nested in it have equi-indices $(2, 3)$ and $(3, 2)$ respectively. □

Theorem 6.1 permits to prove the following Theorems:

Theorem 6.2. *There exists a 4-kite-design of order v and equi-indices* $(2, 3)$*, with v odd, if and only if* :

$$v \equiv 1 \ mod \ 4, \ v \geq 5.$$

Proof The statement follows directly from Theorem 6.1 and considering that the design Σ_5, defined in Z_5 and having for blocks all the translates of the *base* 4-kite:

$$[(2, 1, 3), (0), 4],$$

is a 4-kite-design of order 5 and equi-indices (2,3). □

Theorem 6.3. *For every* $v \equiv 1 \ mod \ 4, \ v \geq 5$, *there exists a strongly balanced 4-kite-design of order v.*

Proof See Theorems 6.1 and 6.2. □

References

[1] L.Berardi, M.Gionfriddo, R.Rota, *Perfect octagon quadrangle systems*, Discrete Mathematics, 310 (2010), 1979–1985.

[2] L.Berardi, M.Gionfriddo, R.Rota, *Perfect octagon quadrangle systems with an upper C_4-system and a large spectrum*, Computer Science Journal of Moldova (Discrete Mathematics), 54 (2010), 303–318.

[3] L.Berardi, M.Gionfriddo, R.Rota, *Perfect octagon quadrangle systems with upper C_4-systems*, Journal of Statistical Planning and Inference, 141 (2011), 2249–2255.

[4] L.Berardi, M.Gionfriddo, R.Rota, *Balanced and strongly balanced P_k-designs*, Discrete Mathematics, to appear.

[5] L.Berardi, M.Gionfriddo, R.Rota, *Perfect octagon quadrangle systems - II*, Discrete Mathematics, to appear.

[6] L.Gionfriddo, *Two constructions for perfect triple systems*, Bull. of ICA, 48 (2006), 73–81.

[7] L.Gionfriddo, *Hexagon quadrangle systems*, Discrete Maths. 309 (2008), 231–241.

[8] L.Gionfriddo, *Hexagon biquadrangle systems*, Australasian J. of Combinatorics 36 (2007), 167–176.

[9] L.Gionfriddo, *Hexagon kite systems*, Discrete Mathematics, 309 (2009), 505–512.

[10] L.Gionfriddo, *Perfect dodecagon quadrangle systems*, Discrete Mathematics, to appear.

[11] S. Kucukcifci, C.C.Lindner, *Perfect hexagon triple systems*, Discrete Maths., 279 (2004), 325–335.

[12] C.C.Lindner, A.Rosa, *Perfect dexagon triple systems*, Discrete Maths. 308 (2008), 214–219.

Luigia Berardi, Mario Gionfriddo, Rosaria Rota Received June 28, 2011

Luigia Berardi
Dipartimento di Ingegneria Elettrica e dell'Informazione,
Universitá di L'Aquila
E–mail: *luigia.berardi@ing.univaq.it*

Mario Gionfriddo
Dipartimento di Matematica e Informatica,
Universitá di Catania
E–mail: *gionfriddo@dmi.unict.it*

Rosaria Rota
Dipartimento di Matematica, Universitá di RomaTre
E–mail: *rota@mat.uniroma3.it*

Computer Science Journal of Moldova, vol.**19**, no.3(57), 2011

A Smooth Newton Method for Nonlinear Programming Problems with Inequality Constraints

Vasile Moraru

Abstract

The paper presents a reformulation of the Karush-Kuhn-Tucker (KKT) system associated nonlinear programming problem into an equivalent system of smooth equations. Classical Newton method is applied to solve the system of equations. The superlinear convergence of the primal sequence, generated by proposed method, is proved. The preliminary numerical results with a problems test set are presented.

Keywords: nonlinear programming, KKT conditions, strict complementarity, Newton method, local superlinear convergence.

1 Introduction

We consider the following nonlinear programming problem:

$$\begin{cases} \text{minimize } f(x) \\ \text{subject to} \\ g_i(x) \geq 0, i = 1, 2, \ldots, m, \end{cases} \tag{1}$$

where $x \in \mathbb{R}^n$, f, $g_i : \mathbb{R}^n \to \mathbb{R}$ are assumed to be twice continuously differentiable. Such problems have proved extremely useful in very many areas of activity, in science, engineering and management [1,2].

Let the *Lagrange function* of problem (1) be defined by

$$L(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i g_i(x),$$

where $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)^T$ is the Lagrange multiplier vector.

We will denote by x^* any local solution of the problem (1), and by

$$\mathcal{A}(x) = \{i : g_i(x) = 0\}$$

the index set of active constraints at x.

The *Karush-Kuhn-Tucker system* (KKT system for short) associated with (1) is [3,4] :

$$\nabla f(x) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0 \quad (\textit{stationarity}) \tag{2}$$

$$g_i(x) \geq 0, i = 1, 2, \ldots, m \quad (\textit{primal feasibility}) \tag{3}$$

$$\lambda_i \geq 0, i = 1, 2, \ldots, m \quad (\textit{dual feasibility}) \tag{4}$$

$$\lambda_i g_i(x) = 0, i = 1, 2, \ldots, m \quad (\textit{complementarity}) \tag{5}$$

This system is the local equivalent to the problem (1) whenever the inequalities (3) and (4) are satisfied and relations (2) and (5) comply with the conditions of regularity [3–5].

We assume that the following hypotheses hold for any local solution x^*.

Assumption A1. The active constraint gradients $\nabla g_i(x^*), i \in \mathcal{A}(x^*)$ are linearly independent (the assumption is called the *linear independence constraint qualification* (LICQ)).

Assumption A2. *Strict complementarity* holds at x^*, i.e. $\lambda_i^* > 0$ for all $i \in \mathcal{A}(x^*)$.

Assumption A3. The *strong second order sufficient condition* (SSOSC):

$$p^T \nabla_{xx}^2 L(x^*, \lambda^*) p \geq c \|p\|^2, c > 0, \text{for all } \; p \in T(x^*), \tag{6}$$

where $\nabla_{xx}^2 L(x^*, \lambda^*)$ is the Hessian matrix of the Lagrange function of the problem (1) and

$$T(x^*) = \left\{ p \in \mathbb{R}^n : p \neq 0, [\nabla g_i(x^*)]^T p = 0, \; i \in \mathcal{A}(x^*) \right\}.$$

It is well know that, under **Assumptions A1–A3**, x^* is an isolated local minimum and to solve (1) is equivalent to solve the KKT system (2) – (5).

KKT system (2) – (5) establishes necessary conditions [5,6] for solving *finite-dimensional variational inequality*:

$$[\nabla f(x^*)]^T (x - x^*) \geq 0, \quad \text{for all } x \in \Omega, \tag{7}$$

where $\Omega = \{x \in \mathbb{R}^n : g_i(x) \geq 0, i = 1, 2, \ldots, m\}$.

In the particular case when $\Omega = \{x \in \mathbb{R}^n : x_i \geq 0, i = 1, 2, \ldots, n\}$ the variational inequality problem (7) is equivalent to the following *complementarity problem*

$$\nabla f(x) \geq 0, x \geq 0, [\nabla f(x)]^T x = 0.$$

Furthermore, the KKT system $(2) - (5)$ can be written as a *mixed complementarity problem* (MCP) [8]:

$$[F(z^*)]^T (z - z^*) \geq 0, \quad \text{for all} \quad z \in B,$$

where $z^T = \left(x^T, \lambda^T\right), B = \{z \in \mathbb{R}^n \times \mathbb{R}^m : \lambda \geq 0\}$ and

$$[F(z)]^T = \left([\nabla_x L(z)]^T, g_1(x), g_2(x), \ldots, g_m(x)\right).$$

The first major achievements obtained in the constrained optimization referred to the KKT systems. This has brought to the development of methods of optimization, a very active area with remarkable results [1–7]. Conditions (5), also called complementarity conditions, raise some difficult problems to solve directly the system of equations and inequalities (2)–(5). Depending on how KKT system (2)–(5) is used (i.e. how Lagrange multipliers are calculated and how the conditions of complementarity are ensured), some methods that can be used successfully in problems with medium or even large number of variables have been developed [9]:

- active-set methods [5,7,9,10],

- barrier/penalty and augmented Lagrangian methods [3,7, 10, 11, 12],
- sequential linear and quadratic programming methods [6,7,13,14],
- interior-point and trust region methods [15–17].

Another way of solving KKT system is to use procedures based on the complementary functions [18,19]. The relationship $(3)-(5)$ constitutes the essence of nonlinear complementarity problems. The KKT system $(2)-(5)$ may be equivalently reformulated as solving the nonlinear system [20, 21]:

$$\nabla_x L(x,\lambda)=0, \varphi(\lambda_i, g_i(x))=0, i=1,2,\ldots,m, \tag{8}$$

where φ is any NCP function. A function $\varphi:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$ is called NCP function if the set of solutions of equation $\varphi(a,b)$ coincides with the set:

$$M=\{a,b\in\mathbb{R}: ab=0, a\geq 0, b\geq 0\}.$$

Classical examples of NCP functions are the min function

$$\varphi(a,b)=\min(a,b)$$

and the Fischer-Burmeister function [22]

$$\varphi(a,b)=\sqrt{a^2+b^2}-a-b.$$

Other functions of complementarity with applications in optimization can be found in the works [23, 24]. Most of them are nondifferentiable at the point $(0,0)$. There are smooth functions of complementarity, for example, the function [23, 30]

$$\varphi(a,b)=2ab-[\min(0,a+b)]^2.$$

This theme in optimization problems still presents challenges, which are based mainly on the application of the Newton method for systems of equations equivalent to system:

$$\nabla_x L(x,\lambda)=0, g_i(x)=0, i\in\mathcal{A}(x^*). \tag{9}$$

Relations (3)−(5) can be reformulated as the system of equalities by introducing auxiliary variables $y_1, y_2, ..., y_m$, so that (see [25,26,31,32]):

$$\begin{cases} [-\min(0,y_i)]^k - g_i(x)=0, \\ [\max(0,y_i)]^k - \lambda_i = 0, \\ i=1,2,\ldots,m, \end{cases} \tag{10}$$

where $k\geq 1$. In [31,32] it is considered $k=1$, in [25] and [26] take $k=2$, respectively $k=3$.

Emphasize the role of regularity condition **A2**. Only in this case we can guarantee that in the vicinity of (x^*,λ^*) relations $(3)-(5)$ are equivalent to the system of equations:

$$\begin{aligned} g_i(x) &= 0, \quad i\in\mathcal{A}(x^*), \\ \lambda_i &= 0, \quad i\notin\mathcal{A}(x^*). \end{aligned}$$

In other words, the **Assumption A2**, the KKT system $(2)-(5)$ is locally equivalent to system of equations (9). Both the procedure (8) and procedure (10) do not require the explicit identification of the active constraints $\mathcal{A}(x^*)$.

In the present paper a KKT system transformation in a system of smooth equations, which can be solved by classical Newton method is considered. The paper is organized as follows. In Section 2 we define the functions that ensure the nonsingularity of Jacobian at a solution. Section 3 presents the algorithm of Newton method. Superlinear convergence in terms of primal variables is proved in Section 4. Some numerical results are given in Section 5 and conclusion is drawn in Section 6.

Throughout this paper, $\mathbb{R}^n$ denotes the space of n-dimensional real column vectors and the superscript "T " denotes transpose. For convenience, we use (x,λ,y) to denote the column vector $\left(x^T,\lambda^T,y^T\right)^T$.

Given two vectors $x, y \in \mathbb{R}^n$, $x^T y$ denotes the Euclidian scalar product and $\|\bullet\|$ denotes the Euclidian vector norm. The identity matrix is denoted by I. For any $\alpha > 0, \beta > 0, \alpha = o(\beta)$ (respectively $\alpha = O(\beta)$) means α/β tends to zero (respectively α/β is uniformly bounded) as $\beta \to 0$.

2 Smoothing Reformulation of the Karush-Kuhn-Tucker System

We define two functions $u, v : \mathbb{R} \to \mathbb{R}_+$ with the following properties:

P1.

$$u(x) = \begin{cases} = 0, & \text{for all } x \leq 0, \\ > 0, & \text{for all } x > 0, \end{cases}$$
$$v(x) = \begin{cases} > 0, & \text{for all } x < 0, \\ = 0, & \text{for all } x \geq 0. \end{cases}$$

P2. $u(x)$ and $v(x)$ is at least twice continuously differentiable on $\mathbb{R}$.

P3. $u(x) = 0$ and $v(x) = 0$ if and only if $u'(x) = 0$, respectively $v'(x) = 0$ for all $x \neq 0$.

The functions $u(x)$ and $v(x)$ so defined form a complementarity pair in the sense that the two functions are complementary to one another (if one is zero at a point, then the other is necessarily nonzero at that point):

$$\begin{cases} u(x) = 0 \Longleftrightarrow v(x) > 0, \\ v(x) = 0 \Longleftrightarrow u(x) > 0, \end{cases} \tag{11}$$

i.e. $u(x) \times v(x) = 0$ for all $x \in \mathbb{R}$.

The following can serve as an example of functions $u(x)$ and $v(x)$ that satisfy properties **P1 - P3**:

$$\begin{cases} u(x) = x^{\beta} \max(0, x) = \frac{1}{2}\left(x^{\beta+1} + |x| x^{\beta}\right), \\ v(x) = (-1)^{\gamma+1} x^{\gamma} \min(0, x) = \frac{(-1)^{\gamma+1}}{2}\left(x^{\gamma+1} - |x| x^{\gamma}\right), \end{cases} \tag{12}$$

where $\beta \geq 2, \gamma \geq 2$ are any fixed parameters.

The functions $u(x)$ and $v(x)$ as defined by the formulas (12) are continuously differentiable:

$$\begin{aligned} u'(x) &= \frac{\beta+1}{2}(x^{\beta} + |x|\, x^{\beta-1}), \\ u''(x) &= \frac{\beta(\beta+1)}{2}(x^{\beta-1} + |x|\, x^{\beta-2}), \end{aligned}$$

$$\begin{aligned} v'(x) &= \frac{(-1)^{\gamma+1}(\gamma+1)}{2}\left(x^{\gamma} + |x|\, x^{\gamma-1}\right), \\ v''(x) &= \frac{(-1)^{\gamma+1}\gamma(\gamma+1)}{2}\left(x^{\gamma-1} + |x|\, x^{\gamma-2}\right). \end{aligned}$$

By entering the auxiliary variables $y_1, y_2, \ldots, y_m$ the KKT system (2) – (5) may be transformed into an equivalent system of smooth nonlinear equations:

$$\begin{cases} \nabla_x L(x,\lambda) = 0, \\ u(y_1) - g_1(x) = 0, \\ \vdots \\ u(y_m) - g_m(x) = 0, \\ v(y_1) - \lambda_1 = 0, \\ \vdots \\ v(y_m) - \lambda_m = 0. \end{cases} \tag{13}$$

It is easily found that $\lambda_i > 0$ for all $i \in \mathcal{A}(x^*)$. Indeed, according to (11) if $g_i(x) = 0$, i.e. $u(y_i) = 0$, then $\lambda_i = v(y_i) > 0$. Therefore $(x^*, \lambda^*) \in \mathbb{R}^n \times \mathbb{R}^m$ solves KKT system (2) – (5) or system (9) if and only if $(x^*, \lambda^*, y^*) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$ solves the system of equations (13).

For any $y = (y_1, y_2, \ldots, y_m)^T \in \mathbb{R}^m$ define

$$U(y) = (u(y_1), u(y_2), \ldots, u(y_m))^T,$$

$$V(y) = (v(y_1), v(y_2), \ldots, v(y_m))^T,$$

$$U'(y) = diag\left(u'(y_1), u'(y_2), \ldots, u'(y_m)\right),$$

$$V'(y) = diag\left(v'(y_1), v'(y_2), \ldots, v'(y_m)\right).$$

Matrices $U'(y)$ and $V'(y)$ are diagonal matrices of dimension $m \times m$ with elements $u'(y_i)$, respectively $v'(y_i)$, $i = 1, 2, \ldots, m$.

We denote the Jacobian matrix of a mapping

$$G(x) = (g_1(x), g_2(x), \ldots, g_m(x))^T : \mathbb{R}^n \to \mathbb{R}^m$$

by $G'(x)$:

$$G'(x) = \begin{pmatrix} [\nabla g_1(x)]^T \\ [\nabla g_2(x)]^T \\ \vdots \\ [\nabla g_m(x)]^T \end{pmatrix}$$

Let $F'(x, \lambda, y)$ denote the Jacobian of

$$F(x, \lambda, y) = \begin{pmatrix} \nabla_x L(x, \lambda) \\ u(y_1) - g_1(x) \\ \vdots \\ u(y_m) - g_m(x) \\ v(y_1) - \lambda_1 \\ \vdots \\ v(y_m) - \lambda_m \end{pmatrix}.$$

Then the Jacobian matrix $F'(x, \lambda, y)$ has the form

$$F'(x, \lambda, y) = \begin{pmatrix} \nabla^2_{xx}(Lx, \lambda) & -[G'(x)]^T & O_{n\times m} \\ -G'(x) & O_{m\times n} & U'(y) \\ O_{m\times n} & -I_m & V'(y) \end{pmatrix},$$

where I_m is the identity matrix of order m and O represents the null matrix, with subscripts indicating their dimensions.

The matrix $F'(x, \lambda, y)$ has order $(n + 2m) \times (n + 2m)$.

The next theorem is true:

Theorem 1. ***Under the Assumption A1-A3 for*** $z^* = (x^*, \lambda^*, y^*)$ ***Jacobian matrix*** $F'(z^*)$ ***is nonsingular.***

Proof. Let $d = (p, q, r) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$ satisfy $F'(z^*) d = 0$. Then

$$\nabla^2_{xx} L(x^*, \lambda^*) p - \left[G'(x^*)\right]^T q = 0, \tag{14}$$

$$-G'(x^*) p + U'(y^*) r = 0, \tag{15}$$

$$-q + V'(y^*) r = 0. \tag{16}$$

By multiplying equation (14) on the left side by p^T we have

$$p^T \nabla^2_{xx} L(x^*, \lambda^*) p = p^T \left[G'(x^*)\right]^T q = q^T \left[G'(x^*)\right]^T p. \tag{17}$$

There is at least an index k such that $u(y^*_k) = 0$ (because otherwise the $g_i(x^*) > 0$ for any $i = 1, 2, \ldots, m$, i.e. x^* is a point of unconstrained minimum). As $g_k(x^*) = u(y^*_k) = 0$ yields $k \in \mathcal{A}(x^*)$. As we have that **Property P3** and $u'(y^*_k) = 0$, which together with (15) gives us

$$[\nabla g_k(x^*)]^T p = 0 \text{ for all } k \in \mathcal{A}(x^*), \tag{18}$$

i.e. $p \in T(x^*)$.

For $s \notin \mathcal{A}(x^*)$ we have $u(y^*_s) \neq 0$ and so $v(y^*_s) = 0$ where the $v'(y^*_s) = 0$; then from the (16) result $q_s = 0$. This together with the relationship (18) gives us $q^T G'(x^*) p = 0$. From equation (16) together with the **Assumption A3** we have $p = 0$. So, from equations (14) – (16) one obtains

$$\sum_{i=1}^{m} q_i \nabla g_i (x^*) = 0, \tag{19}$$

$$u' (y_i^*) r_i = 0 \quad \text{for all } i = 1, 2, \ldots, m, \tag{20}$$

$$q_i = v' (y_i^*) r_i \quad \text{for all } i = 1, 2, \ldots, m. \tag{21}$$

We have $\lambda_i^* = 0, v (y_i^*) = 0, v' (y_i^*) = 0, u' (y_i^*) = 0$ for all $i \notin \mathcal{A} (x^*)$. From (20) and (21) $q_i = 0, r_i = 0$, for all $i \notin \mathcal{A} (x^*)$. It is true that

$$\sum_{i \in \mathcal{A}(x^*)} q_i \nabla g_i (x^*) = 0,$$

where, under the **Assumption A1**, $q_i = 0$ for all $i \in \mathcal{A} (x^*)$. For $i \in \mathcal{A} (x^*)$ we have $v' (y_i^*) \neq 0$ and from (21) it results that $r_i = 0$. Thus, the **Assumption A1-A3** implies that $p = 0, q = 0, r = 0$. Therefore, the Jacobian matrix $F' (x^*, \lambda^*, y^*)$ is nonsingular. ■

3 Local Newton method

The best-known method for solving nonlinear systems of equations is the Newton method [27,28,29]. Newton's method has very attractive theoretical and practical properties, because of its rapid convergence: under the nonsingularity of the Jacobian matrix it will converge locally superlinearly. If in addition, the Jacobian is Lipshitz continuous, then the convergence is quadratic.

Let $\left(x^{(0)}, \lambda^{(0)}, y^{(0)}\right)$ be given sufficiently close to (x^*, λ^*, y^*). Given that $x^{(k)}$, $\lambda^{(k)}$, $y^{(k)}$ is the k-th iterate of the Newton method for the system (13), a new estimate $x^{(k+1)}$, $\lambda^{(k+1)}$, $y^{(k+1)}$ is defined by solving the following linear system:

$$\begin{cases} \nabla^2_{xx}(Lx^{(k)}, \lambda^{(k)})(x - x^{(k)}) - \left[G'\left(x^{(k)}\right)\right]^T (\lambda - \lambda^{(k)}) = \\ \qquad = -\nabla_x(Lx^{(k)}, \lambda^{(k)}), \\ -G'(x^{(k)})(x - x^{(k)}) + U'(y^{(k)})(y - y^{(k)}) = \\ \qquad = -U(y^{(k)}) + G(x^{(k)}), \\ -\lambda + \lambda^{(k)} + V'(y^{(k)})(y - y^{(k)}) = -V(y^{(k)}) + \lambda^{(k)}. \end{cases} \tag{22}$$

The system (22) consists of $(n + 2m)$ equations and $(n + 2m)$ unknowns. From the last equation of the system (22) we have

$$\lambda = V(y^{(k)}) + V'(y^{(k)})(y - y^{(k)}). \tag{23}$$

Substituting (23) in the first equation, the system (22) becomes

$$\begin{cases} \nabla^2_{xx} L\left(x^{(k)}, \lambda^{(k)}\right)\left(x - x^{(k)}\right) - \left[G'\left(x^{(k)}\right)\right]^T V'\left(y^{(k)}\right)\left(y - y^{(k)}\right) = \\ \qquad = -\nabla f\left(x^{(k)}\right) + \left[G'\left(x^{(k)}\right)\right]^T V\left(y^{(k)}\right), \\ -G'\left(x^{(k)}\right)\left(x - x^{(k)}\right) + U'\left(y^{(k)}\right)\left(y - y^{(k)}\right) = \\ \qquad = -U\left(y^{(k)}\right) + G\left(x^{(k)}\right). \end{cases} \tag{24}$$

The system (24) is from $(n + m)$ equations with $(n + m)$ unknowns.

According to Theorem 1, in the neighborhood of z^* the system of equations (24) admits the unique solution $\left(x^{(k+1)}, y^{(k+1)}\right)$.

On the other hand, we see that system (13) can be transformed into an equivalent system which only contains x and y:

$$\begin{cases} \nabla_x L\left(x, V(y)\right) = 0, \\ -G\left(x\right) + U\left(y\right) = 0. \end{cases} \tag{25}$$

Linearizing the system (25), we obtain the same system of linear equations (24), except that in the Hessian matrix $\nabla^2_{xx} L\left(x^{(k)}, \lambda^{(k)}\right)$ we have $\lambda^{(k)} = V\left(y^{(k)}\right)$.

From (23) it follows that

$$\begin{aligned}\lambda^{(k+1)} &= V\left(y^{(k)}\right) + V'\left(y^{(k)}\right)\left(y^{(k+1)} - y^{(k)}\right) = \\ &= V\left(y^{(k+1)}\right) + o\left(\left\|y^{(k+1)} - y^{(k)}\right\|\right).\end{aligned}$$

So, both local direct linearization of the system (13) and linearization of the system (25) are close enough. The difference may appear only if you are not near the solution.

Thus we can define the following algorithm.

Algorithm 1. ***Local version of Newton's method.***

Step 1. *Let* $x^{(0)} \in \mathbb{R}^n$, $y^{(0)} \in \mathbb{R}^m$, $\lambda^{(0)} = V\left(y^{(0)}\right)$, $\varepsilon > 0$ and $k = 0$.

Step 2. *If*

$$\max\left\{\left\|\left[G'(x^{(k)})\right]^T V(y^{(k)}) - \nabla(fx^{(k)})\right\|, \left\|U(y^{(k)}) - G(x^{(k)})\right\|\right\} < \varepsilon,$$

STOP.

Otherwise, let $x^{(k+1)} \in \mathbb{R}^n, y^{(k+1)} \in \mathbb{R}^m$ be a solution of linear system (24).

Step 3. *Let the multiplier vector*

$$\lambda^{(k+1)} = V(y^{(k)}) + V'(y^{(k)})\left(y^{(k+1)} - y^{(k)}\right),$$

and $k = k + 1$.

Go to Step 2.

Remark 1. The **Algorithm 1** generates a sequence of pairs $\left(x^{(k)}, y^{(k)}\right)$ – the solution of system of linear equations (24) which consists of $(n + m)$ equations with $(n + m)$ unknowns. The Lagrange multiplier λ is determined as a function of y through the formula (23). Therefore the algorithm can be considered a primal-dual method.

Remark 2. After reformulation, **Algorithm 1** can be used for solving the complementarity problems and variational inequality problems.

Remark 3. As it is well known, Newton method possesses just local convergence, it is very sensitive to the choice of initial approximations and is not convergent if it is not sufficiently close to the solution. There are several ways to modify the method to ensure its global convergence:

- Damped Newton's method [27],
- The Levenberg-Marquardt scheme,
- Trust-regions approach [5,28].

4 Primal superlinear convergence

Newton's method for solving system of linear equations (24) has the advantage of high convergence of couple $\left(x^{(k)}, y^{(k)}\right)$. In the following we show that the rate of convergence for the sequence of primal variables $\left\{x^{(k)}\right\}$ is also superlinear.

In addition to the **Properties P1-P3**, we assume that the functions $u(x)$ and $v(x)$ satisfy the following **Property P4**:

P4.

$$u(x) \times v'(x) = u'(x) \times v(x) = 0 \quad \text{for all } x \in \mathbb{R}.$$

It is not difficult to see that the functions from example (12) also satisfy the **Property P4**.

Theorem 2. ***Let us suppose that the standard Assumptions A1-A3 are satisfied. Assume also that functions*** $u(x)$ ***and*** $v(x)$ ***satisfy the Properties P1-P4. Then the sequence*** $\left\{x^{(k)}\right\}$ ***generated by Algorithm 1 for problem (1) converges locally to*** x^* ***superliniarly.***

Proof. The linear system (24) together with (23) are equivalent to the system (22). Let $x^{(k+1)}$, $\lambda^{(k+1)}$ and $y^{(k+1)}$ be solution for the system equations (22). It is easy to follow the relations obtained from the first equation of system (22):

$$-\nabla_{xx}^2 L\left(x^{(k)},\lambda^{(k)}\right)\left(x^{(k+1)}-x^{(k)}\right)=\nabla_x L\left(x^{(k)},\lambda^{(k+1)}\right)=$$

$$=\nabla_x L\left(x^*,\lambda^{(k+1)}\right)+\left[\nabla_x L\left(x^{(k)},\lambda^{(k+1)}\right)-\nabla_x L\left(x^*,\lambda^{(k+1)}\right)\right]=$$

$$=\nabla_x L\left(x^*,\lambda^{(k+1)}\right)+\nabla_{xx}^2 L\left(x^*,\lambda^{(k+1)}\right)\left(x^{(k+1)}-x^*\right)+$$

$$+o\left(\left\|x^{(k)}-x^*\right\|\right)=\nabla_x L\left(x^*,\lambda^*\right)+\nabla_{x\lambda}^2 L\left(x^*,\lambda^*\right)\left(\lambda^{(k+1)}-\lambda^*\right)+$$

$$=\nabla_x L\left(x^*,\lambda^*\right)+\nabla_{x\lambda}^2 L\left(x^*,\lambda^*\right)\left(\lambda^{(k+1)}-\lambda^*\right)+$$

$$+\nabla_{xx}^2 L\left(x^*,\lambda^{(k+1)}\right)\left(x^{(k+1)}-x^*\right)+o\left(\left\|x^{(k)}-x^*\right\|\right)=$$

$$=\left[G'\left(x^*\right)\right]^T\left(\lambda^{(k+1)}-\lambda^*\right)+\nabla_{xx}^2 L\left(x^*,\lambda^*\right)\left(x^{(k+1)}-x^*\right)+$$

$$=\left[\nabla_{xx}^2 L\left(x^*,\lambda^{(k+1)}\right)-\nabla_{xx}^2 L\left(x^*,\lambda^*\right)\right]\left(x^{(k)}-x^*\right)+$$

$$+o\left(\left\|x^{(k)}-x^*\right\|\right)=\left[G'\left(x^*\right)\right]^T\left(\lambda^{(k+1)}-\lambda^*\right)+$$

$$+\nabla_{xx}^2 L\left(x^*,\lambda^*\right)\left(x^{(k+1)}-x^*\right)-\nabla_{xx}^2 L\left(x^*,\lambda^*\right)\left(x^{(k+1)}-x^{(k)}\right)+$$

$$+o\left(\left\|x^{(k)}-x^*\right\|\right),$$

where we have

$$\left[\nabla^2_{xx}L\left(x^*,\lambda^*\right)-\nabla^2_{xx}L\left(x^{(k)},\lambda^{(k)}\right)\right]\left(x^{(k+1)}-x^*\right)=\left[G'\left(x^*\right)\right]^T\times \tag{26}$$

$$\times\left(\lambda^{(k+1)}-\lambda^*\right)+\nabla^2_{xx}L\left(x^*,\lambda^*\right)\left(x^{(k+1)}-x^*\right)+o\left(\left\|x^{(k)}-x^*\right\|\right).$$

Taking into consideration the **Properties P1-P3** from the last equation of system (22) and from (23), one obtains:

$$\lambda_i^{(k+1)}g_i\left(x^{(k+1)}\right)=\left[v\left(y_i^{(k)}\right)+v'\left(y_i^{(k)}\right)\left(y_i^{(k+1)}-y_i^{(k)}\right)\right]\times$$

$$\times g_i\left(x^{(k+1)}\right)+\left[\nabla g_i\left(x^{(k+1)}\right)\right]^T\left(x^{(k+1)}-x^{(k)}\right)+\left(x^{(k+1)}-x^{(k)}\right)=$$

$$=\left[v\left(y_i^{(k)}\right)+v'\left(y_i^{(k)}\right)\left(y_i^{(k+1)}-y_i^{(k)}\right)\right]\times$$

$$\times\left[u\left(y_i^{(k)}\right)+u'\left(y_i^{(k)}\right)\left(y_i^{(k+1)}-y_i^{(k)}\right)\right]+o\left(\left\|x^{(k)}-x^*\right\|\right).$$

So,

$$\lambda_i^{(k+1)}g_i\left(x^{(k+1)}\right)=o\left(\left\|x^{(k)}-x^*\right\|\right)\quad\text{for any } i. \tag{27}$$

On the other hand,

$$\lambda_i^{(k+1)}g_i\left(x^{(k+1)}\right)=\lambda_i^*g_i\left(x^*\right)+g_i\left(x^*\right)\left(\lambda_i^{(k+1)}-\lambda_i^*\right)+$$

$$+\lambda_i^* \left[\nabla g_i\left(x^*\right)\right]^T \left(x^{(k+1)} - x^*\right) + o\left(\left\|x^{(k+1)} - x^*\right\|\right).$$

From here and from (27), taking into consideration the **Assumption A2**, we have for all $i \in \mathcal{A}\left(x^*\right)$:

$$\left[\nabla g_i\left(x^*\right)\right]^T \left(x^{(k+1)} - x^*\right) = -\delta_i^{(k)}, \tag{28}$$

where

$$-\delta_i^{(k)} = o\left(\left\|x^{(k+1)} - x^*\right\|\right) + o\left(\left\|x^{(k+1)} - x^{(k)}\right\|\right).$$

As the system of vectors $\{\nabla g_i\left(x^*\right)\}$, $i \in \mathcal{A}\left(x^*\right)$, is linearly independent (**Assumption A2)**, there is a vector $\gamma^{(k)} \in \mathbb{R}^n$ such that:

$$\left[\nabla g_i\left(x^*\right)\right]^T \gamma^{(k)} = \delta_i^{(k)} \tag{29}$$

and

$$\left\|\gamma^{(k)}\right\| = o\left(\left\|x^{(k+1)} - x^*\right\|\right) + o\left(\left\|x^{(k+1)} - x^{(k)}\right\|\right).$$

Let now

$$p^{(k)} = x^{(k+1)} - x^* + \gamma^{(k)} \in \mathbb{R}^n.$$

Then (28) and (29) shows that

$$\left[\nabla g_i\left(x^*\right)\right]^T p^{(k)} = 0 \quad \text{for all } i \in \mathcal{A}\left(x^*\right),$$

i.e. $p^{(k)} \in T\left(x^*\right)$. We also notice that

$$\left[p^{(k)}\right]^T \left[G'\left(x^*\right)\right]^T \left(\lambda^{(k+1)} - \lambda^*\right) = 0. \tag{30}$$

Indeed

$$\left[p^{(k)}\right]^T \left[G'\left(x^*\right)\right]^T \left(\lambda^{(k+1)} - \lambda^*\right) =$$

$$= \sum_{i \in \mathcal{A}(x^*)} \left(\lambda_i^{(k+1)} - \lambda_i^*\right) \left[\nabla g_i\left(x^*\right)\right]^T p^{(k)} = 0.$$

From (29) and (30) we have

$$\left[p^{(k)}\right]^T \left[\nabla^2_{xx} L\left(x^*, \lambda^*\right) - \nabla^2_{xx} L\left(x^{(k)}, \lambda^{(k)}\right)\right] \left(x^{(k+1)} - x^*\right) =$$

$$= \left[p^{(k)}\right]^T \nabla^2_{xx} L\left(x^*, \lambda^*\right) \left(x^{(k+1)} - x^*\right) + o\left(\left\|x^{(k+1)} - x^*\right\|\right) \left\|p^{(k)}\right\|.$$

Finally, the last relationship together with **Assumption A2**, gives us

$$c \left\|p^{(k)}\right\|^2 \leq \left[p^{(k)}\right]^T \nabla^2_{xx} L\left(x^*, \lambda^*\right) p^{(k)} =$$

$$\left[p^{(k)}\right]^T \left[\nabla^2_{xx} L\left(x^*, \lambda^*\right) - \nabla^2_{xx} L\left(x^{(k)}, \lambda^{(k)}\right)\right] \left(x^{(k+1)} - x^*\right) +$$

$$+ \left[p^{(k)}\right]^T \nabla^2_{xx} L\left(x^*, \lambda^*\right) \gamma^{(k)} + o\left(\left\|x^{(k)} - x^*\right\|\right) \left\|p^{(k)}\right\| =$$

$$= o\left(\left\|x^{(k+1)} - x^*\right\|\right) \left\|p^{(k)}\right\| + O\left(\left\|p^{(k)}\right\| \left\|\gamma^{(k)}\right\|\right) + o\left(\left\|x^{(k)} - x^*\right\|\right) \times$$

$$\times \left\|p^{(k)}\right\| = o\left(\left\|x^{(k+1)} - x^*\right\|\right) \left\|p^{(k)}\right\| + o\left(\left\|x^{(k)} - x^*\right\|\right) \left\|p^{(k)}\right\|,$$

where

$$\left\|p^{(k)}\right\| = o\left(\left\|x^{(k+1)} - x^*\right\|\right) + o\left(\left\|x^{(k)} - x^*\right\|\right).$$

So,

$$\left\|x^{(k+1)} - x^*\right\| = \left\|p^{(k)} - \gamma^{(k)}\right\| \leq \left\|p^{(k)}\right\| + \left\|\gamma^{(k)}\right\| =$$

$$= o\left(\left\|x^{(k+1)} - x^*\right\|\right) + o\left(\left\|x^{(k)} - x^*\right\|\right).$$

The last relationship implies that

$$\left\|x^{(k+1)} - x^*\right\| = o\left(\left\|x^{(k)} - x^*\right\|\right),$$

This completes the proof.■

5 Test examples

In this section, we give some numerical results. The algorithm described in this paper was implemented by a Maple code and tested on a selection of problems from [2] and [33]. As functions $u(x)$ and $v(x)$, there were taken the concrete functions (12), where $\beta = \gamma = 2$.

Example 1. (**Problem:16 [33, p. 39]**).

$$\begin{cases} f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \to \min \\ \text{s.t.} \quad x_1 + x_2^2 \geq 0, \quad x_1^2 + x_2 \geq 0, \\ -2 \leq x_1 \leq 0.5, \quad x_2 \leq 1. \end{cases}$$

The starting points: $x^{(0)} = (-2, 1)$, $y^{(0)} = (-1, 1)$.
The optimal solution:

$$\begin{aligned} x^* &= (0.5,\ 0.25),\ f(x^*) = 0.25, \mathcal{A}(x^*) = (4), \\ y^* &= (0.82548, 0.7937, 1.3572, -1.0, 0.90856), \\ \lambda^* &= (0.0,\ 0.0,\ 0.0, 1.0,\ 0.0). \end{aligned}$$

Example 2. (**Problem:43 [33, p. 66]**).

$$\begin{cases} f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \to \min \\ \text{s.t.} \quad 8 - x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 \geq 0, \\ 10 - x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 \geq 0, \\ 5 - 2x_1^2 - x_2^2 - x_3^2 - x_4^2 - 2x_1 + x_2 + x_4 \geq 0. \end{cases}$$

The starting points: $x^{(0)} = (0, 0, 0, 0)$, $y^{(0)} = (-1, 1, -1)$.
The optimal solution:

$$\begin{aligned} x^* &= (0.135\,90,\ 1.092\,7, 1.857\,8,\ -1.138\,7\ ),\ f(x^*) = -43.716, \\ y^* &= (-1.010\,3, 0.817\,48, -1.103\,4), \\ \lambda^* &= (1.031\,1,\ 0.0,\ 1.343\,3),\ \mathcal{A}(x^*) = (1, 3) \end{aligned}$$

Example 3. (Problem:64 [33, p. 86]).

$$\begin{cases} f(x) = 5x_1 + \frac{50000}{x_1} + 20x_2 + \frac{72000}{x_2} + 10x_2 + \frac{144000}{x_3} \to \min \\ \text{s.t. } 1 - \frac{4}{x_1} - \frac{32}{x_2} - \frac{120}{x_3} \geq 0, \quad 0.00001 \leq x_1 \leq 300, \\ 0.00001 \leq x_2 \leq 300, \quad 0.00001 \leq x_3 \leq 300. \end{cases}$$

The starting points: $x^{(0)} = (1, 1, 1)$, $y^{(0)} = (-1, 1, -1, 1, 1, 1, 1)$.
The optimal solution:

$$\begin{aligned} x^* &= (108.73,\ 85.126,\ 204.32),\ f(x^*) = 6299.8424, \\ y^* &= (-13.160,\ 4.773\,0,\ 5.761\,6,\ 4.399,\ 5.989\,6,\ 5.889\,9,\ 4.573\,7), \\ \lambda^* &= (2279.0,\ 0,\ 0,0,\ 0,0,0),\ \mathcal{A}(x^*) = (1). \end{aligned}$$

6 Conclusion

We have presented and tested smoothing Newton methods for solving nonlinear optimization problems, with requirements compared to those of the classical Newton method. The basic idea of the methods is to replace the KKT system by a system which is appropriate and equivalent to the one on which we apply Newton's method. It is shown that by reasonable choice of functions $u(x)$ and $v(x)$ we obtain effective methods for solving the nonlinear programming problems. The methods are simple and can be applied with different functions $u(x)$ and $v(x)$ that satisfy the **Properties P1-P4.** The numerical results show that the proposed method produces fast local convergence. From the practical point of view, the possibility to relax the assumptions of strict complementarity conditions (**Assumption A2**) remains an open question.

References

[1] N.I.M. Gould, Ph.L Toint, *How mature is nonlinear optimization?* In Applied mathematics entering the 21st century: ICIAM 2003 Congress, SIAM, Philadelphia (2004), pp. 141–161.

[2] N. Andrei, *Models, Test Problems and Applications for Mathematical Programming.* Technical Press, Bucharest, 2003 (in Romanian).

[3] D.P. Berstsekas, *Nonlinear programming.* 2nd edition, Athena Scientific, Belmont, Massachusetts, 1999.

[4] M.S. Bazaraa, H. D. Sheraly,C.M. Shetty *Nonlinear programming. Theory and algorithms.* 3 rd. edition, Wiley Interscience. A John Wiley & Sons, Inc. Publication, 2006.

[5] R. Fletcher, *Practical methods of optimization.* Second edition, on Wiley & Sons, New-York, 1990 (republished in paperback 2000).

[6] B.N. Pshenichnyi, Yu.M. Danilin, *Numerical methods in extremal problems.* Mir, Moscow, 1978 (translated from the Russian).

[7] J. Nocedal, S.J. Wright, *Numerical optimization.* Second edition, Spriner Verlag, New-York, 1998.

[8] F. Facchinei, J.S. Pang, *Finite-Dimensional variational inequalities and complementarity problems.* Springer-Verlag, New-York, 2003.

[9] N.I.M. Gould, D. Orban, Ph.L Toint, *Numerical methods for large-scale nonlinear optimization.* Acta Numerica, Cambridge University Press (2005), pp. 299–361.

[10] Ph. E. Gill, W. Murray, M.H. Wright, *Practical optimization.* Second edition, Academic Press Inc., 1982.

[11] A.V. Fiacco, G.P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques.* Classics in Applied Mathematics, SIAM, Philadelphia, PA, second edition, 1990.

[12] R.T. Rockafellar, *Lagrange multipliers and optimality.* SIAM Review, 35, pp. 183–238.

[13] P.T. Boggs, J.W. Tolle, *Sequential quadratic programming.* Acta Numerica, Cambridge University Press (1995), pp. 1–51.

[14] N.I.M. Gould, Ph.L Toint, *SQP methods for large-scale nonlinear programming.* In System Modelling and Optimization, Methods, Theory and Applications. Kluwer Academic Publishers, (2000), pp. 149–178.

[15] S.J. Wright, *Primal-dual interior-point methods.* SIAM, Philadelphia, 1997.

[16] A. Forsgren, Ph. E. Gill, Wright, M.H. Wright, *Interior methods for nonlinear optimization.* SIAM, Review, 44 (2002), pp.525–597.

[17] A. Nemirovski, M.J.Todd, *Interior-point methods for optimization.* Acta Numerica, Cambridge University Press (2009), pp. 1–44.

[18] A. Fischer, *An NCP-Function and use for the solution of complementarity problems.* Recent Advandces in Nonsmooth Optimization (1995), pp. 88–105.

[19] M. C. Ferris,Ch. Kanzow, *Complementarity and related problems: A survey.* In Handbook of Applied Optimization. Oxford University Press, New-York (2002), pp. 514–530.

[20] Ch. Kanzow, H. Kleinmichel, *A class of Newton-type methods for equality and inequality constrained optimization.* Optimization Methods and Software, 5, (2) (1995), pp.173–198.

[21] Z.-H. Huang, D. Sun, G. Zhao, *A smootting Newton-type algorithm of stronger convergence for the quadratically constrained quadratic programming.* Computational Optimization and Applications, 35 Issue (2), (2006) 5, (2), pp.199–237.

[22] A. Fischer, *An special Newton type optimization method.* Optimization, 24, (1992), pp. 269–284.

[23] Y.G. Evtushenco, V.A. Purtov, *Sufficient conditions for a minimum for nonlinear programming problems.* Soviet Mathematics Doklady, 30, (1984), pp. 313–316.

[24] A.F. Izmailov, M.V. Solodov, *Numerical Methods of Optimization.* Fizmatlit/Nauka, Moscow, Russia, Second Edition – 2008. (In Russian).

[25] O. Stein, *Lifting mathematical programs with complementary constraints.* Mathematical Programming, 29, (2010), DOI: 10.1007/s10107-010-0345-y.

[26] A.F. Izmailov, A.L. Pogosyan, M.F. Solodov, *Semismooth Newton method for the lifted reformulation of mathematical programs with complementarity constraints mathematical programs with complementary constraints.* Computational Optimization and Application, (2010), DOI: 10.1007/s10589-010-9341-7.

[27] J.M. Ortega, W.C. Rheinboldt, *Iterative solution of nonlinear equation in several variables.* Academic Press, New-York, 1970.

[28] J.E. Dennis, R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice Hall, Englewood Cliff, 1983.

[29] T.J. Ypma, *Historical development of the Newton-Raphson method.* SIAM Review, 37 (4), (1995), pp. 531–551.

[30] C. Kanzow, *Some equation – based methods for nonlinear complementarity problem.* Optimization Method and Software, 3 (1), (1994), pp. 327-340.

[31] M. Kojima, *Strongly stable stationary solution in nonlinear programs.* In S. M. Robinson (Ed), Analysis and Computation of Fixed Points, Academic Press, New-York, (1980), pp. 93–138.

[32] J.-B.Hiriart-Urruty, *Optimisation et analyse convexe.* Presses Universitaires de France, Paris, 1998.

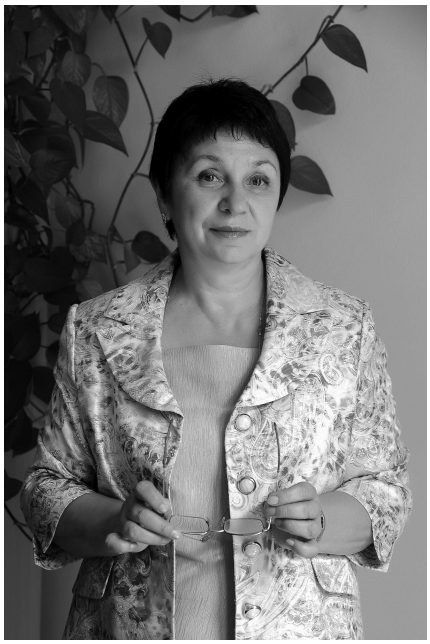
[33] W. Hock, K. Schittkowski, *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer-Verlag, Berlin / Heidelberg / New York, 1981.

Vasile Moraru

Received April 9, 2011

Technical University of Moldova,
Faculty of Computers, Informatics and Microelectronics
Bd. Stefan cel Mare, 168, MD-2004, Chisinau, Moldova
Phone: (373-2) 50-99-01
E–mail: *moraru@mail.utm.md*

Director of IMCS – National Prize Laureate of Moldova



Director of the Institute of Mathematics and Computer Science, Vice Editor-in-Chief of CSJM, and our colleague, D.Hab. Svetlana Cojocaru, in 2011 became the National Prize Laureate of Moldova. In accordance with Government decision, this distinction is given for "outstanding achievements whose results have substantially enriched science, culture and art, had a considerable contribution to promoting a positive image of the country in the international arena, a significant impact on the development of socio-economic, scientific and technical progress, national and world culture."

D.Hab. Svetlana Cojocaru contributed to the development and promotion of information systems capable of dealing with various Information Society issues, including facilitating "human-computer" interaction both by developing intelligent interfaces in general and in particular, by using natural language, computer applications development for medicine, art, etc. She solved the problem of automatic morphological inflection for Romanian, which permitted to create a large lexical data base with a lot of useful applications.

Our congratulations and all the best wishes for the future!

Editorial board of the "Computer Science Journal of Moldova"