A Preliminary Study of the Application of the P-graph Methodology for Organization-based Multiagent System Designs: Assessment

Juan C. García-Ojeda[1,2], Botond Bertok[2], Ferenc Friedler[2], Andres Argoti[3] and L. T. Fan[3]

[1]Department of Systems Engineering, Autonomous University of Bucaramanga, Av. 42 No 48-11, El Jardín, Bucaramanga, Santander 680003, Colombia, E-mail: jgarciao@unab.edu.co

[2]Department of Computer Science and Systems Technology, University of Pannonia, Egyetem u. 10, H-8200 Veszprém, Hungary, E-mail: {bertok, friedler}@dcs.vein.hu

[3]Department of Chemical Engineering, Kansas State University, 1005 Durland Hall, Manhattan, Kansas 66506, U. S. A. E-mail:{argoti,fan}@ksu.edu

Abstract: At the outset, the design of an organization-based multiagent system is modeled according to the framework of Organization Model for Adaptive Complex Systems (OMACS). Subsequently, this design model is transformed into a process-network model. Eventually, the resultant process-network model in conjunction with the P-graph-based methodology give rise to: (i) the maximal structure of the process network, comprising all the feasible combinations of structures, i.e., OMACS-based design configurations, capable of yielding the specified products from the specified raw material; (ii) every feasible structure for the process of interest; and (iii) the optimal structure of the network, i.e., the optimal OMACS-based design configuration. Finally, in light of the tenet of a modeling-transformation-evaluation paradigm, an appraisal is made of the feasibility as well as the flexibility and cost of the optimal OMACS-based design configuration obtained.

Keywords: Organization-based Multiagent System Design; Model Transformation; Process Synthesis; P-graph Framework; OMACS Framework

1 Introduction

Designing and implementing large, complex, and distributed systems by resorting to autonomous or semi-autonomous agents that can reorganize themselves by cooperating with one another represent the future of software systems [4]. A set of methodologies [21], a selection of design processes [3], and a collection of frameworks [4], [5], [6], [7], [8], [16], [23], [27], [35], [36] are available in the literature to provide the basis for constructing sophisticated autonomous multiagent organizations. Moreover, a set of metrics and methods have been suggested with the intention of providing useful information about key properties (e.g., complexity, flexibility, self-organized, performance, scalability, and cost) of these multiagent organizations [24], [26], [31], [33].

The above-mentioned methodologies, processes, frameworks, and procedures, nevertheless, do not offer techniques for identifying the number of feasible configurations[1] that can be synthesized, or designed, from a set of heterogeneous agents. This is an important issue when designing a multiagent system because of the nature of the environments where it operates (dynamic, continuous, and partially accessible) [29]. The multiagent system must be adaptive (self-organized) to adjust its behavior to cope with the dynamic appearance and disappearance of goals (tasks), their given guidelines, and the overall goal of the multiagent system [29], [30]. To address such an issue, i.e., identifying feasible agent´s configurations, we propose a novel approach based upon previous work on organization-based multiagent systems [4] and the P-graph methodology [10].

The current contribution describes the deployment of the P-graph methodology for synthesizing organization-based multiagent systems based upon the OMACS framework. The remainder of the current contribution comprises: an outline of the OMACS framework (Section 2); a brief description of the P-graph methodology (Section 3); the motivational example for undertaking the work (Section 4); a procedure for transforming an organization-based multiagent system design into a process-synthesis problem (Section 5); a description of the mathematical programming model for synthesizing organization-based multiagent systems (Section 6); the preliminary results of the proposed approach (Section 7); and finally, the conclusions as well as the proposed future work (Section 8).

2 The Framework of Organization Model for Adaptive Computational Systems: OMACS

The Framework of Organization Model for Adaptive Computational Systems (hereafter, OMACS) defines the entities in standard multiagent systems and their relationship as a tuple $O_{OMACS} = \langle G_{OMACS}, R_{OMACS}, A_{OMACS}, C_{OMACS}, \Phi, P_{OMACS}, \Sigma,$ *oaf, achieves, capable, requires, possesses, potential*$\rangle$, and it is also represented via an UML[2]-based organizational meta-model (see Figure 1) [4]. These are briefly described in what follows.

[1] These feasible configurations may be seen as agent´s organizations or agents' teams [4].

[2] Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering.

The organization, O_{OMACS}, is composed of four entities including G_{OMACS}, R_{OMACS}, A_{OMACS}, and C_{OMACS}. G_{OMACS} defines the goals of the organization (i.e., overall functions of the organization); R_{OMACS} defines a set of roles (i.e., positions within an organization whose behavior is expected to achieve a particular goal or set of goals). A_{OMACS} is a set of agents, which can be either human or artificial (hardware or software) entities that perceive their environment (Σ – domain model) and can perform actions upon it. In order to perceive and to act, the agents possess a set of capabilities (C_{OMACS}), which define the percepts/actions at their disposal. Capabilities can be soft (i.e., algorithms or plans) or hard (i.e., hardware related actions). P_{OMACS} formally specifies rules that describe how O_{OMACS} may or may not behave in particular situations.

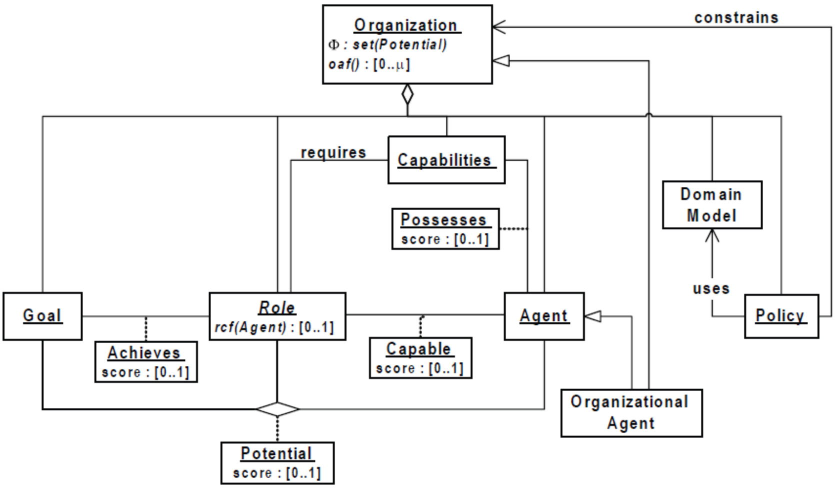

Figure 1

OMACS Meta-model [4]

In addition, OMACS defines a set of functions – *achieves*, *requires*, *possesses capable*, *potential*, *oaf*, and ϕ – to capture the different relations among the entities. *achieves*, a function whose arguments are a goal in G_{OMACS} as well as a role in R_{OMACS} that generates an output which is a positive real number greater than or equal to 0 and less than or equal to 1 (*achieves*, R_{OMACS} x $G_{OMACS} \rightarrow [0,1]$, defines the extent of achievement of a goal by a role); *possesses*, a function with an agent in A_{OMACS} and a capability in C_{OMACS} as inputs yields a positive real number in the range of [0,1] (possesses, A_{OMACS} x $C_{OMACS} \rightarrow [0,1]$, defines the quality of an agent´s capability); requires, a function that assumes a role in R_{OMACS}, thereby yielding a set of capabilities required to play that role (*requires*, $R_{OMACS} \rightarrow \rho(C_{OMACS})$, defines the set of capabilities required to play a role[3]); *capable*, a function whose inputs are an agent in A_{OMACS} and a role in R_{OMACS} and generates an

[3] ρ denotes power set.

output, which is a positive real number greater than or equal to 0 and less than or equal to 1 (*capable*, A_{OMACS} x R_{OMACS} → [0,1], defines how well an agent can play a role), thus giving rise to

$$capable(a,r) = \begin{cases} 0 & if \prod_{c \in requires(r)} possesses(a,c) = 0 \\ \frac{\sum_{c \in requires(r)} possesses(a,c)}{|requires(r)|} & elsewhere; \end{cases} \quad (1)$$

potential, a function with an agent in A_{OMACS}, a role in R_{OMACS}, and a goal in G_{OMACS} as inputs yields a positive real number in the range of [0,1], thus yielding

$$potential(a, r, g) = achieves(r, g) * capable(a, r) \quad (2)$$

(*potential*, A_{OMACS} x R_{OMACS} x G_{OMACS} → [0,1], defines how well an agent can play a role to achieve a goal), and assignment set, ϕ, the set of agent-role-goal tuples $<a, r, g>$, indicating that agent $a \in A_{OMACS}$ has been assigned to play role $r \in R_{OMACS}$ in order to achieve goal $g \in G_{OMACS}$ (ϕ is a subset of all the potential assignments of agents to play roles to achieve goals). Finally, the selection of ϕ from the set of potential assignments is defined by the organization's reorganization function, *oaf*, that assumes a set of assignments in ϕ, thereby yielding a positive real number in the range of [0,∞] (*oaf*, $\rho(\phi)$ → [0,∞], defines the quality of a proposed set of assignments, i.e., *oaf* computes the goodness of the organization based on ϕ), thus resulting in

$$oaf = \sum_{<a_i, r_j, g_j> \in \phi_{OMACS}} potential(a_i, r_j, g_j). \quad (3)$$

3 Process Network Synthesis

In a process system, raw materials are consumed through various transformations (e.g., chemical, physical, and biological) to desired products. Vessels where these transformations take place are called operating units of the process. A given set of operating units with likely interconnections can be portrayed as a network.

The desired products can be also manufactured via some sub-networks of the above-mentioned network. Thus, a given network may give rise to a variety of processes, or process networks, producing the desired products, and each of such process networks corresponds to a sub-network, that can be considered regarded as its structure. Energy and raw material consumption strongly depend on the selection of a process structure; thus, the optimal design of such a process structure, i.e., the process network synthesis (PNS), or process synthesis in short, has both environmental and economic implications [14].

A number of methods has been developed for process synthesis [28]. These methods can be classified according to whether they are based on heuristics or algorithms, i.e., mathematical programming approaches. The majority, if not all, of these methods, however, may not be sufficiently effective for PNS of a realistic or industrial scale, process because of its combinatorial complexity arising from the involvement of a large number of interconnected loops [14]. To cope with this, an innovative approach based on P-graphs (process graphs), which are unique, mathematically rigorous bipartite graphs, has been proposed to facilitate the process network synthesis [10]. The P-graphs are capable of capturing not only the syntactic but also semantic contents of a process network. Subsequently, an axiom system underlying the P-graph framework is constructed to define the combinatorial feasible process-network structures. The analysis and optimization of properties of such structures are performed by a set of efficient combinatorial algorithms: MSG [9], SSG [9], and ABB [13].

3.1 Process Graph (P-graph)

The mathematical definition of a P-graph and a process structure represented by it are elaborated below [10].

Finite set *M*, containing materials, and finite set *O*, containing operating units, are given such that

$$0 \subseteq \rho(M)\ x\ \rho(M) \tag{4}$$

Thus, a P-graph can be defined to be a pair, *(M,O)*, as follows:

The vertices of the graph are the elements of

$$V = M\ x\ O \tag{5}$$

Those belonging to set *M* are of the *M*-type vertices, and those belonging to set *O* are of *O*-type vertices. The arcs of the graph are the elements of

$$A = A_1 \cup A_2 \tag{6}$$

where

$$A_1 = \{(X,Y) \mid Y = (\alpha,\beta) \epsilon\ O \text{ and } X \epsilon\ \alpha\} \tag{7}$$

and

$$A_2 = \{(Y,X) \mid Y = (\alpha,\beta) \epsilon\ O \text{ and } X \epsilon\ \beta\ \} \tag{8}$$

In these expressions, *X* designates an *M*-type vertex; *Y*, an *O*-type vertex; α a set of *M*-type vertices from which arcs are directed into the *O*-type vertices; and, β a set of *M*-type vertices to which arcs are directed out of the *O*-type vertices.

For illustration let *M* be a set of materials, *M={A,B,C,D,E,F}*, and *O* be a set of operating units given by *O={({B,A},{A}),({D,E},{B,C}),({F},{A,C}), ({F},{A,C})}*. It is not difficult to validate that sets *M* and *O* satisfies constraint (1), i.e., *(M,O)* is a P-graph, as depicted in Figure 2.

3.2 Solution Structures

The materials and operating units in a feasible process structure must always conform to certain combinatorial properties. For example, a structure containing no linkage between a raw material and a final product is unlikely to represent any practical process. Hence, it is of vital importance to identify the general combinatorial properties to which a structure must conform. More important, the properties identified should be satisfied by the structure of any feasible solution of the synthesis problem. In other words, those and only those structures satisfying these properties can be feasible structures of a process: no other structures or constraints need to be considered in synthesizing the process.

A set of axioms has been constructed to express necessary and sufficient combinatorial properties to which a feasible process structure should conform. Next, each axiom is stated: (S1) Every final product is represented in the graph; (S2) A vertex of the *M*-type has no input if and only if it represents a raw material; (S3) Every vertex of the *O*-type represents an operating unit defined in the synthesis problem; (S4) Every vertex of the *O*-type has at least one path leading to a vertex of the *M*-type representing a final product; and, (S5) If a vertex of the *M*-type belongs to the graph, it must be an input to or output from at least one vertex of the *O*-type in the graph.

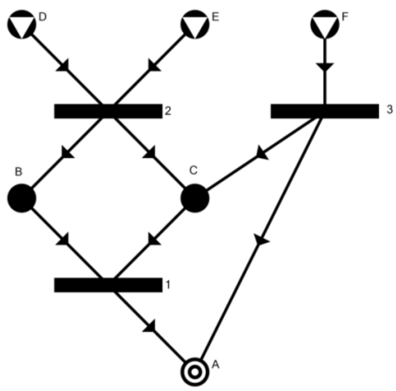

Figure 2

P-graph *(M,O)* where *A*,*B*,*C*,*D*,*E*, and *F* are materials, and *1*,*2*, and *3* are the operating units: represents raw materials or input elements of the whole process; symbolizes intermediate-materials or elements, emerging between the operating units; and represents products or outputs of the entire process

If a P-graph of a given synthesis problem, *(P,R,O)*[4], satisfies theses axioms, it is defined to be a solution-structure of the problem.

4 Where $P \subseteq M$ is the set of product, $R \subseteq M$ is the set of raw materials, and O the set of operating units.

3.3 Algorithms MSG, SSG, and ABB

Both the P-graph representation of a process network and the set of five axioms for solution structures, i.e., combinatorial feasible networks, render it possible to fashion the three mathematically rigorous algorithms: MSG, SSG, and ABB. The algorithm MSG (Maximal-Structure Generation) generates the maximal structure (super-structure) of a process synthesis network. Also, the algorithm SSG (Solution-Structure Generation) generates the set of feasible process structures from the maximal structure, which leads to the algorithm ABB (Accelerated Branch and Bound) for computing the n-best optimal solution structure [9], [10], [11], [12], [13].

4 Motivational Example: Cooperative Robotic Search Team System

To demonstrate the application of the P-graph framework for assessing the design of OMACS-based multiagent systems, a survey is given of a simplified Cooperative Robotic Search Team[5] (CRST) system [20], [33]. Essentially, we are to design a team of robots whose goal is to search for different areas of a given location on a map. The team should be able to search any area of the given location even when faced with failures of individual robots or specific capabilities of those robots. This implies that the team must be able to: (1) assign areas based on individual team member´s reliability; (2) recognize when a robot is unable to perform adequately its duties; and (3) reorganize the team to allow it to achieve its goals in spite of individual failures.

4.1 Overview of CRST Organization

For illustration, it is presumed that four goals be achieved by the CRST. In other words, $G = \{g_1, g_2, g_3, g_4\}$ where g_i for $1 \leq i \leq 4$ signifies "search area *i*." In the CRST, two roles are identified, i.e., $R = \{r_1,r_2\}$ where r_1 and r_2 represent the Searcher and Patroller roles, respectively. In particular, role $r_1$requires the Sonar, Movement, and GPS capabilities for achieving goals g_1, g_2, g_3, and g_4. Likewise, role r_2 requires the Movement, GPS, and Range Finder capabilities for achieving the same goals as those of role r_1. Moreover, for each goal, g_j, an achieve value is assigned. This achieve value defines the extent of achievement of a goal by a role. Both, the *requires* and *achieves* relations can be formally stated as: *requires* $= \{(r_1,\{c_1, c_2, c_3\}),(r_2,\{c_2, c_3, c_4\})\}$ and *achieves* $= \{(r_1, g_1, 0.2), (r_1, g_2, 0.4), (r_1, g_3, 0.8), (r_1, g_4, 1.0), (r_2, g_1, 1.0), (r_2, g_2, 0.7), (r_2, g_3, 0.4), (r_2, g_4, 0.1)\}$.

5 Although the CRST presented in this paper has been simplified (due to space constraint), it is still interesting enough to illustrate our work.

Also, four capabilities are specified. They are Sonar (c_1), Movement (c_2), GPS (c_3), and Range Finder (c_4). c_1 captures information about all objects around agent a_i (in a 360° view). c_2 allows agent a_i to move in any direction, north, south, east, or west (up, down, left, or right). c_3 provides the ability to read the absolute position of agent a_i in the environment. Finally, c_4 renders it possible for agent a_i to measure the distance of the closest object directly in front of it.

In addition, three different agents are modeled; they are a_1, a_2, and a_3. Specifically, agent a_1 possesses capabilities c_1, c_2, c_3, and c_4 while both agents a_2 and a_3 possess capabilities c_2, c_3, and c_4. The possesses relationship is formulated as follows: *possesses* = $\{(a_1,c_1,0.3), (a_1,c_2,0.5), (a_1,c_3,0.3), (a_1,c_4,0.3), (a_2,c_2,0.7), (a_2,c_3,0.5), (a_2,c_4,0.7), (a_3,c_2,0.4), (a_3,c_3,0.9), (a_3,c_4,0.2)\}$.

Additionally, the cost of each individual agent a_1, a_2, and a_3 is \$850, \$900, and \$950, respectively (see Figure 3).

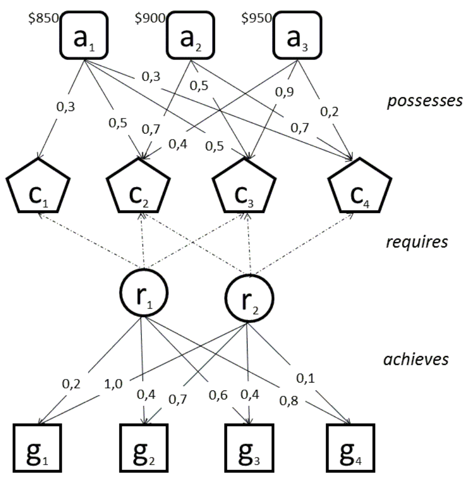

Figure 3

Overview of the CRST Organization. The boxes at the top of the diagram represent agents identified by their types, circles represent the roles, pentagons represent capabilities, and squares are system's goals. The arrows between the entities represent functions/relations *achieves*, *requires*, and *possesses*

5 Algorithm OMACStoPNS

Algorithm OMACStoPNS comprises two mayor parts, the initialization and the construction parts. The initialization part (statements *st1*, *st2*, *st3*, and loop *lp1*) specifies the sets of available raw materials and desired products to be manufactured as well as their parameters. The construction part (loop *lp3*) specifies the set of candidates operating units as well as their parameters (see Figure 3).

input: $G_{OMACS}, A_{OMACS}, R_{OMACS}, achieves, requires, possesses$
comment: G_{OMACS} defines the goals of the organizations, R_{OMACS} defines a set of roles, A_{OMACS} is a set of agents, $achieves$ defines the extent of achievement of a goal by a role, $(G_{OMACS} \times R_{OMACS} \rightarrow [0 \dots 1])$ $possesses$ defines the quality of an agent´s capability $(A_{OMACS} \times C_{OMACS} \rightarrow [0 \dots 1])$, and $requires$ defines the set of capabilities required to play a role $(R_{OMACS} \rightarrow \wp(C_{OMACS}))$. The $capable$ function $(A_{OMACS} \times R_{OMACS} \rightarrow [0 \dots 1])$
output: sets P, R, O
comment: $R \subset M, P \subset M, R \cap P = \emptyset$
begin
comment: initialization part of the algorithm;
st1: $M := M \cup \{oaf\}$;
st2: $P := P \cup \{oaf\}$;
st3: $U_{oaf} := \infty; L_{oaf} := 0; c_{oaf} := 1$;
lp1: **for** $a_i \in A_{OMACS}$ **do**
begin
$R := R \cup \{a_i\}; M := M \cup \{a_i\}; U_{a_i} := \infty; L_{a_i} := 0; c_{a_i} := cost(a_i)$;
end;
comment: construction part of the algorithm;
lp2: **for** $g_i \in G_{OMACS}$ **do**
begin
$M := M \cup \{g_i\}; g_{i\_oaf} := \left\{\{g_i\}, \{oaf\}\right\}; O := O \cup \{g_{i\_oaf}\}$;
$U_{g_{i\_oaf}} := \infty; L_{g_{i\_oaf}} := 0; c_{g_{i\_oaf}} := 0$;
$a_{g_i, g_{i\_oaf}} := 1; a_{g_{i\_oaf}, oaf} := 1$;
end;
lp3: **for** $a_i \in A_{OMACS}$ **do**
begin
$Capabilities_{a_i} := \emptyset$;
for $\langle a', c, value' \rangle \in possesses$ **do**
begin
if $a' = a_i$ **then**
$Capabilities_{a_i} := Capabilities_{a_i} \cup \{c\}$;
end;
end;
for $\langle r_k, \wp(c) \rangle \in requires$ **do**
begin
if $\wp(\mathrm{c}) \subseteq Capabilities_{a_i}$ **then**
$aux := \emptyset$;
for $\langle r'', g_j, value'' \rangle \in achieves$ **do**
begin
if $r_k = r''$ **then**
$M := M \cup \{a_i\_r_k\_g_j\};\ a_i\_r_k\_g_j := \left\{\{a_i\_r_k\_g_j\}, \{g_j\}\right\}$;
$aux := aux \cup \{a_i\_r_k\_g_j\};\ O := O \cup \{a_i\_r_k\_g_j\}$;
$U_{a_i\_r_k\_g_j} := \infty; L_{a_i\_r_k\_g_j} := 0; c_{a_i\_r_k\_g_j} := 0$;
$a_{a_i\_r_k\_g_j, a_i\_r_k\_g_j} := 1; a_{a_i\_r_k\_g_j, g_j} := value''$;
$a_{a_i, a_1\_r_k} = 1; a_{a_i\_r_k, a_i\_r_k\_g_j} := capable(a_i, r_k)$;
end;
if *aux is not empty* **then**
$a_i\_r_k := \left\{\{a_i\}, \{aux\}\right\}; O := O \cup \{a_i\_r_k\}$;
$U_{a_i\_r_k} := 1; L_{a_i\_r_k} := 0; c_{a_i\_r_k} := 0$;
end;
end;
end;

Figure 4
Algorithm OMACStoPNS written in Pidgin Algol

Each agent a_i in A_{OMACS}, is transformed into raw material r and added to set R (loop *lp1*); as such, Axiom (S2) is satisfied. Algorithm OMACStoPNS generates the resources, R_{a_i}; R_{a_1}, R_{a_2}, and R_{a_3}. Furthermore, lower bound L_{a_i}, upper bound

U_{a_i}, and cost c_{a_i}, are set for each resource, R_{a_i}; as such, algorithm OMACStoPNS specifies the total amount of available resources for the motivational problem (see Table 1). Thus, only a single product, *oaf*, is specified and added to set *P* (statements *st1*, *st2*, and *st3*); as such, Axiom (S1) is automatically satisfied. Note that this is analogous to the notion of the goodness of the organization based on the quality of a proposed set of assignments. In other words, the set of agent-role-goal tuples $<a_i, r_k, g_j>$ indicates that agent $a_i \epsilon A_{OMACS}$ has been assigned to play role $r_k \epsilon R_{OMACS}$ in order to achieve goal $g_j \epsilon G_{OMACS}$. For outcome *oaf*, algorithm OMACStoPNS sets lower bound L_{oaf}, upper bound U_{oaf}, and cost c_{oaf}; as such, the amount of product to be manufactured for meeting the demand of the problem is specified (see Table 2).

Table 1
Resources to be considered in process synthesis for the example

Resource R_j	**Lower bound R_j**	**Upper bound R_j**	**Cost c_j**
a_1	0	∞	0
a_2	0	∞	0
a_3	0	∞	0

Table 2
Targets to be considered in process synthesis for the example

Target P_j	**Lower Bound L_j**	**Upper Bound U_j**	**Cost c_j**c$_j$
oaf	0	∞	1

Subsequently, algorithm OMACStoPNS systematically specifies the operating units in loops *lp2* and *lp3*, representing organizational assignments, as described in section 3; as such, Axioms (S3) and (S4) are satisfied. First, the algorithm loops through every goal $g_j \epsilon G$. Each goal g_j is transformed into material *m* for inclusion in set *M*. Algorithm OMACStoPNS generates materials M_{g_j}; M_{g_1}, M_{g_2}, M_{g_3}, and M_{g_4}. Note that material M_{g_j} represents the goals to be accomplished by the organization. This gives rise to the creation of operating unit *o* for inclusion in set *O* for each g_j. Algorithm OMACStoPNS generates operating units $Og_{j\_}oaf$. Additionally, lower bound $L_{g_j\_oaf}$, upper bound $U_{g_j\_oaf}$, cost $c_{g_j\_oaf}$, and a_{ji}, the consumption rate of entity m_j by operating unit o_i are set for each operating unit $O_{g_j\_oaf}$; as such, algorithm specifies the goals to be achieved by the system (see Table 1).

Afterwards, algorithm OMACStoPNS loops through every agent $a_i \in A$. Consequently, for each agent a_i, algorithm OMACStoPNS checks whether a_i is capable of playing a given role r_k in *R*. If so, algorithm OMACStoPNS searches for every g_j in *G*, such that g_j is achieved by r_k. As a result, algorithm OMACStoPNS generates materials $M_{a_i\_r_k\_g_j}$; $M_{a_1\_r_1\_g_1}$, $M_{a_1\_r_1\_g_2}$, $M_{a_1\_r_1\_g_3}$, $M_{a_1\_r_1\_g_4}$, $M_{a_1\_r_2\_g_1}$, $M_{a_1\_r_2\_g_2}$, $M_{a_1\_r_2\_g_3}$, $M_{a_1\_r_2\_g_4}$, $M_{a_2\_r_2\_g_1}$, $M_{a_2\_r_2\_g_2}$, $M_{a_2\_r_2\_g_3}$, $M_{a_2\_r_2\_g_4}$,

$M_{a_3\_r_2\_g_1}$, $M_{a_3\_r_2\_g_2}$, $M_{a_3\_r_2\_g_3}$, and $M_{a_3\_r_2\_g_4}$. Subsequently, for each agent a_i, role r_k, and goal g_j, two operating units o are created and added to set O. One indicates that agent a_i is capable of playing role r_k; the second implies that agent a_i has been assigned to play role r_k in order to achieve goal g_j. Accordingly, algorithm OMACStoPNS generates the operating units $O_{a_i\_r_k}$ and $O_{a_i\_r_k\_g_j}$. Moreover, lower bounds $L_{a_i\_r_k}$ and $L_{a_i\_r_k\_g_j}$; upper bounds $U_{a_i\_r_k}$ and $U_{a_i\_r_k\_g_j}$; and costs $c_{a_i\_r_k}$ and $c_{a_i\_r_k\_g_j}$; and the consumption flow rate of material m_j, a_{ji}, by operating unit o_i, are set for each of operating units $O_{a_i\_r_k}$ and $O_{a_i\_r_k\_g_j}\mathrm{O}_{\mathrm{a_i\_r_k\_g_j}}$; as such, algorithm specifies whether agent a_i has been assigned to play role r_k in order to achieve goal g_j (see Table 3). As a result, the execution of loop *lp3* assures that Axiom (S5) is satisfied by the maximal structure. Figure 5 displays the maximal structure of the motivational example generated by algorithm MSG.

Table 3

Operating units to be considered in process synthesis for the example*

Operating Unit O_i	**Input Material m_j**	**Output Material m_j**	**Lower bound L_i**	**Upper bound U_i**	**Cost c_i**
g_1_oaf	g_1 (1)	oaf (1)	0	∞	0
g_2_oaf	g_2 (1)	oaf (1)	0	∞	0
g_3_oaf	g_3 (1)	oaf (1)	0	∞	0
g_4_oaf	g_4 (1)	oaf (1)	0	∞	0
$a_1\_r_1\_g_1$	$a_1\_r_1\_g_1(0.433)$	$g_1(0.2)$	0	∞	0
$a_1\_r_1\_g_2$	$a_1\_r_1\_g_2(0.433)$	$g_2(0.4)$	0	∞	0
$a_1\_r_1\_g_3$	$a_1\_r_1\_g_3(0.433)$	$g_3(0.6)$	0	∞	0
$a_1\_r_1\_g_4$	$a_1\_r_1\_g_4(0.433)$	$g_4(0.8)$	0	∞	0
$a_1\_r_2\_g_1$	$a_1\_r_2\_g_1(0.433)$	$g_1(1.0)$	0	∞	0
$a_1\_r_2\_g_2$	$a_1\_r_2\_g_2(0.433)$	$g_2(0.7)$	0	∞	0
$a_1\_r_2\_g_3$	$a_1\_r_2\_g_3(0.433)$	$g_3(0.4)$	0	∞	0
$a_1\_r_2\_g_4$	$a_1\_r_2\_g_4(0.433)$	$g_4(0.1)$	0	∞	0
$a_2\_r_2\_g_1$	$a_2\_r_2\_g_1(0.633)$	$g_1(1.0)$	0	∞	0
$a_2\_r_2\_g_2$	$a_2\_r_2\_g_2(0.633)$	$g_2(0.7)$	0	∞	0
$a_2\_r_2\_g_3$	$a_2\_r_2\_g_3(0.633)$	$g_3(0.4)$	0	∞	0
$a_2\_r_2\_g_4$	$a_2\_r_2\_g_4(0.633)$	$g_4(0.1)$	0	∞	0
$a_3\_r_2\_g_1$	$a_3\_r_2\_g_1(0.5)$	$g_1(1.0)$	0	∞	0
$a_3\_r_2\_g_2$	$a_3\_r_2\_g_2(0.5)$	$g_2(0.7)$	0	∞	0
$a_3\_r_2\_g_3$	$a_3\_r_2\_g_3(0.5)$	$g_3(0.4)$	0	∞	0
$a_3\_r_2\_g_4$	$a_3\_r_2\_g_4(0.5)$	$g_4(0.1)$	0	∞	0
$a_1\_r_1$	a_1	$a_1\_r_1\_g_1(0.433)$, $a_1\_r_1\_g_2(0.433)$, $a_1\_r_1\_g_3(0.433)$, $a_1\_r_1\_g_4$ (0.433)	0	1	0

$a_1\_r_2$	a_1	$a_1\_r_2\_g_1(0.433)$, $a_1\_r_2\_g_2(0.433)$, $a_1\_r_2\_g_3(0.433)$, $a_1\_r_2\_g_4$ (0.433)	0	1	0
$a_2\_r_2$	a_2	$a_2\_r_2\_g_1(0.633)$, $a_2\_r_2\_g_2(0.633)$, $a_2\_r_2\_g_3(0.633)$, $a_2\_r_2\_g_4$ (0.633)	0	1	0
$a_2\_r_2$	a_3	$a_3\_r_2\_g_1(0.5)$, $a_3\_r_2\_g_2(0.5)$, $a_3\_r_2\_g_3(0.5)$, $a_3\_r_2\_g_4$ (0.5)	0	1	0

* The numbers in the brackets are the flow rates, a_{ji}, of the input and output materials relative to the unit capacity of each operating unit.

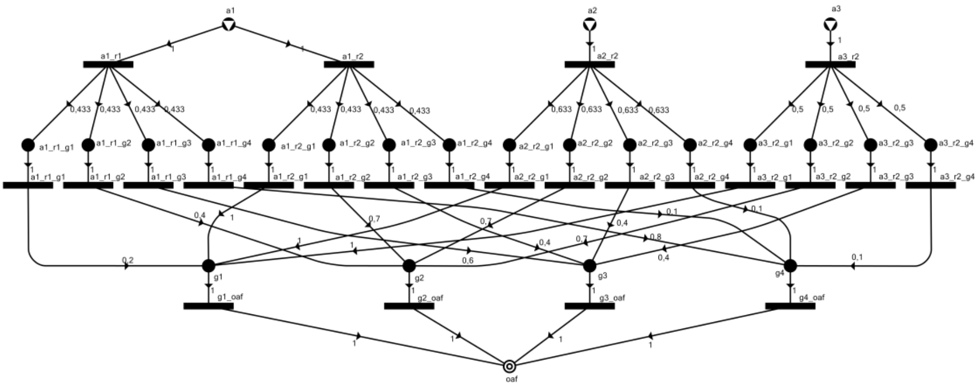

Figure 5

Maximal structure for the hypothetical example to illustrate the solution-structure generation with algorithm MSG

6 Mathematical Programming Model

Unlike any of the available algorithmic methods for computing the quality of a proposed set of assignments based upon OMACS, i.e., agents, $a_i \in A_{OMACS}$, assigned to play roles, $r_k \in R_{OMACS}$, in order to achieve goals, $g_j \in G_{OMACS}$, where no mathematical programming model is derived due to the approach adopted, i.e., step-by-step computation [4], [20], [30], [33], [37], [38]; we propose a simple mathematical programming model, which is derived from the maximal structure, generated by algorithm MSG, and does not impair the optimality of the resultant solution.

In the present work, a mixed-integer linear programming (MILP) model has been formulated, which at the very least yields a solution identical with those conventional OMACS-based assignment algorithms [37], [38].

Let M denote the set of entities; P, the set of products, where $P \subseteq M$; R, the set of initially available resources, where $R \subseteq M$; and O, the set of activities, where $O = \rho(M) \times \rho(M)$. The relations between entities and activities are denoted by a_{ji}

which gives the difference between the production and consumption rate of entity M_j by activity O_i, where $M_j \in M$ and $O_i \in O$. Also given are lower bound L_{O_i} and upper bound U_{O_i} for the volume of each activity O_i, as well as its cost c_{O_i}. In addition, lower bound L_{R_j} and upper bound U_{R_j} are specified for each resource R_j. In addition, lower bound L_{P_j} and upper bound U_{P_j} are defined for each product P_j. Moreover, two classes of variables are involved in the mathematical programming model. One class consists of binary variables, each denoted by $y_{O_i} \in \{0,1\}$ expressing the absence (0) or the existence (1) of operating unit O_i; and the other, continuous variables, each denoted by x_{O_i} expressing the size or capacity of operating unit O_i relative to the unit size. If operating unit O_i is included in the network, as indicated by $y_i = 1$, the concomitant continuous variable, x_{O_i}, can be any real value in the range of 0 to the upper limit for the capacity of operating unit O_i. Thus, $x_{O_i} \le y_{O_i} U_i$, where U_i is the upper limit for the capacity; if such an upper limit does not exist, the U_i can be any large number L. Finally, z, maximal, is the objective value. The resultant MILP model is given in the following.

$$z = max\left(\sum_{P_j \in M \cap P}\left(c_{P_j} * \sum_{O_i \in O} a_{ij} * x_{O_i}\right)\right) \tag{9}$$

subject to (10)

$$M = \bigcup_{(\alpha_i,\beta_i) \in O} \alpha_i \cup \beta_i$$

$$0 \le x_{O_i}, L_{O_i} \le x_{O_i} \le U_{O_i} \qquad \forall O_i \in O \tag{11}$$

$$L_{P_j} \le \sum_{O_i \in O} a_{ij} * x_{O_i} \le U_{P_j} \qquad \forall P_j \in M \cap P \tag{12}$$

$$L_{R_j} \le \sum_{O_i \in O} a_{ji} * x_{O_i} \le U_{R_j} \qquad \forall R_j \in M \cap R \tag{13}$$

$$L_{M_j} \le \sum_{O_i \in O} a_{ji} * x_{O_i} - \sum_{O_i \in O} a_{ij} * x_{O_i} \le U_{M_j} \qquad \forall M_j \in M \setminus (R \cup P) \tag{14}$$

$$x_{O_i} \le y_{O_i} L \tag{15}$$

$$y_{O_i} \in \{0,1\} \tag{16}$$

The maximal structure serves as the input to the generation and solution of the MILP model by algorithm ABB [13]. It yields the optimal network and a finite number of *n*-best suboptimal networks in ranked order. Algorithm ABB has identified a total of 65535 structures[6,7] in less than 75 seconds on an Intel(R) Core(TM) i5 CPU @ 3.20 GHz. Table 4 shows 10 feasible solutions for the example. Algorithms MSG and ABB have been executed by software PNS Studio [32].

Table 4
Subset of Feasible Solutions (less than 1%) generated by algorithm

Sol. #	agent's organization assignment set, ϕ	oaf value	Agents' cost ($)
1	$\left\{\begin{matrix}\langle a_1,r_1,g_1\rangle,\langle a_1,r_1,g_2\rangle,\langle a_1,r_1,g_3\rangle,\langle a_1,r_1,g_4\rangle, \\ \langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle, \\ \langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\end{matrix}\right\}$	4,3112	2700
1280	$\left\{\begin{matrix}\langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle, \\ \langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\end{matrix}\right\}$	3,4452	2700
3204	$\left\{\begin{matrix}\langle a_1,r_1,g_1\rangle,\langle a_1,r_1,g_2\rangle,\langle a_1,r_1,g_3\rangle,\langle a_1,r_1,g_4\rangle, \\ \langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle\end{matrix}\right\}$	3,2112	1750
7813	$\left\{\begin{matrix}\langle a_1,r_1,g_1\rangle,\langle a_1,r_1,g_2\rangle,\langle a_1,r_1,g_3\rangle,\langle a_1,r_1,g_4\rangle, \\ \langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\end{matrix}\right\}$	2,9186	1800
19883	$\left\{\begin{matrix}\langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle, \\ \langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\end{matrix}\right\}$	2,4926	1850
25400	$\left\{\begin{matrix}\langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle\end{matrix}\right\}$	2,3452	1750
36779	$\left\{\begin{matrix}\langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle, \\ \langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\end{matrix}\right\}$	2,0526	1800

[6] It is important to point out that only 77% of the structures, i.e., 50626, are feasible assignments for the problem. OMACS model imposes that a feasible assignment set is based on the current set of goals required to be achieved by the system [4]. For example, assignment set $\phi = \{<a_1,r_1,g_1>,<a_1,r_1,g_3>,<a_2,r_2,g_4>,<a_3,r_2,g_4>\}$ is a valid assignment; however it is unfeasible for the motivational example: goal g_2 will never be achieved.

[7] Algorithm SSG has identified 65535 structures in 4.662 s without computing the optimal and sub-optimal assignments.

45654	$\begin{Bmatrix} \langle a_1,r_1,g_1\rangle,\langle a_1,r_1,g_2\rangle,\langle a_1,r_1,g_3\rangle,\langle a_1,r_1,g_4\rangle, \\ \langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle \end{Bmatrix}$	1,8186	850
57730	$\{\langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle\}$	1,3926	900
62333	$\{\langle a_3,r_2,g_1\rangle,\langle a_3,r_2,g_2\rangle,\langle a_3,r_2,g_3\rangle,\langle a_3,r_2,g_4\rangle\}$	1,1	950

7 Assessment of Organization-based Multiagent System Designs

To empirically evaluate the flexibility of the different agent-based organization designs identified by algorithm ABB, we have developed a simulation that steps through the CRST application. To measure the flexibility, the approach deployed in [33] is followed; specifically, capability failure has been simulated. At each step in the simulation, a randomly selected system goal, i.e., g_1, g_2, g_3, and g_4, is achieved. Subsequently, the best available assignment is calculated (see Eq. 2). The best assignment defines how well an agent, $a_i \in A_{OMACS}$, can play a role (see Eq. 1), $r_k \in R_{OMACS}$, to achieve a goal, $g_j \in G_{OMACS}$. Afterwards, one of the capabilities possessed by a robot is randomly selected and tested to see if it has failed. A predefined capability failure rate (0 – 100%) indicates if the selected capability has failed. Once failed, a capability is assumed to remain so for the life of the system. In addition, reorganization is performed to assign available robots to available goals and to de-assign robots if their capabilities have failed, and thus, they are no longer able to play their assigned roles.

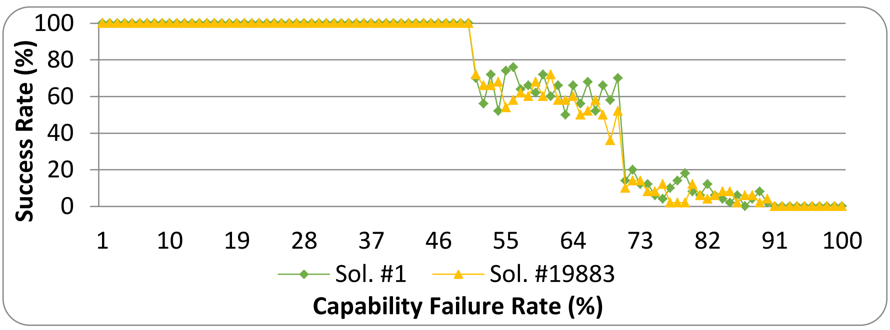

Figure 6

Comparison of Sol. #1 and Sol. # 19883

Each agent-based organization (see Section 6) has been simulated for failure rates ranging from 0 to 100% for 1000 system executions. The comparison of Figures 6 and 7 reveals a difference among the agent-based organization configurations, thereby rendering it possible to offer significant remarks about the claim, "the higher the organization score (i.e., the oaf function), the better the performance of the organization."[36]. First, it is not always the rule that the higher the oaf function score, the better the performance of the agent-based organization. For instance, Figure 6 displays a scenario where an agent-based organization, i.e., Sol. # 19883, with an oaf value of ϕ = 2.4926 and the cost of $1850 performing equally well when compared to the best agent organization, i.e., Sol. # 1, with an oaf value of ϕ = 4.3112 and the cost of $2700.

Moreover, Figure 7 demonstrates another scenario where an agent-based organization, i.e., Sol. #7183, with an oaf value of ϕ = 2.9186 and the cost of $1800, is outperformed[8] by other agent-based organizations, i.e., Sol. #25400 and Sol. #57730, with oaf values of ϕ = 2.3452 and ϕ = 1.3926 whose costs are $1800 and $900, respectively.

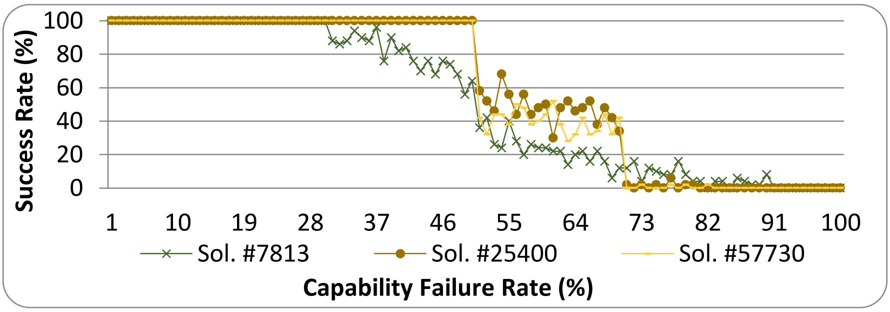

Figure 7
Comparison of Sol. #7813, Sol. #25400, and Sol. #57730

Conclusions and Future Work

In this work we have introduced an algorithmic method for assessing the n-best organizational-based multiagent system design based upon the OMACS framework. The method has been crafted by transforming an organizational-based multiagent system design into a *PNS* problem and solving the resultant problem by the algorithms and the software of the P-graph framework.

The potential of the proposed method has been illustrated by solving an example in which the optimal and suboptimal organizational-based multiagent system designs in ranked order emerge by defining its cost (in terms of the oaf function, i.e., ϕ; see Eq. 3), as the objective function. However, an optimal solution does not always capture the expected behavior of the organizational-based multiagent system design. Thus, additional research is needed to explore the combinations of the P-graph framework with other techniques (e.g., system reliability [16], self-organization [25], complex systems [34], genetic algorithms [1]), which can

[8] This behavior emerges when the capability failure rate ranges from 30% through 70%.

effectively capture the expected behavior of an organizational-based multiagent system in the design phase.

Finally, we propose the construction of a computational tool for transforming OMACS organizational-based multiagent systems into PNS problems and integrating it into existing tools [17], [18]. Our efforts will be the subject of future contributions in this research area; as well, its application in other domains [2], [15].

Acknowledgments

The first author would like to thank Virag Varga for her kind support in the definition of the mathematical model.

In memoriam, Prof. L. T. Fan (1929-2014)

References

[1] Arezki Mellal, M., Adjerid, S., Benazzouz, D., Berrazouane, S., Edward J. Williams, E. J. Optimal Policy for the Replacement of Industrial Systems Subject to Technological Obsolescence – Using Genetic Algorithm. Acta Polytechnica Hungarica, 10(1), 197-208 (2013)

[2] Barrera-Sanabria, G., Arenas-Seleey, D., García-Ojeda, Juan C., Méndez-Ortiz, F. Designing Adaptive Educational Web Sites: General Framework. In Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICALT 2004, pp. 973-977, IEEE Computer Society, Joensuu, Finland (2004)

[3] Cossentino, C., Hilaire, V., Molesini, A., Seidita, V.: Handbook on Agent-Oriented Design Processes. An IEEE-FIPA Standard Compliant Description Approach. Springer-Verlag, Berlin (2014)

[4] DeLoach, S. A., Oyenan, W. H., Matson, E. T.: A Capabilities-based Model for Artificial Organizations. Journal of Autonomous Agents and Multiagent Systems. 16(1), 13-56 (2008)

[5] Dignum, V., Vázquez-Salceda, J., Dignum, F.: Omni: Introducing Social Structure, Norms and Ontologies into Agent Organizations. In: Bordini, R. H. *et al.* (eds.) PROMAS 2004, LNAI 3346, pp. 181-198, Springer-Verlag, Berlin Heildeberg (2005)

[6] Dignum, V. A.: Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD Dissertation, Utrecht University (2004)

[7] Ferber, J., Gutknecht, O.: A Meta-Model for the Analysis and Design of Organizations in Multiagent Systems. In Proceedings of the 3rd International Conference on Multi Agent Systems, pp. 128-135, IEEE Computer Society, Washington, DC (1998)

[8] Fortino, G., Russo, W.: ELDAMeth: An Agent-oriented Methodology for Simulation-based Prototyping of Distributed Agent Systems. Information and Software Technology. 54(6), 608-624 (2012)

[9] Friedler, F., Tarján, K., Huang, Y. W. and Fan, L. T.: Combinatorial Algorithms for Process Synthesis. Computers Chem. Engng. 16, S313-320 (1992)

[10] Friedler, F., Tarján, K., Huang, Y. W. and Fan, L. T.: Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems. Chem. Engng. Sci. 47, 1972-1988 (1992)

[11] Friedler, F., Tarján, K., Huang, Y. W., Fan, L. T.: Graph-Theoretic Approach to Process Synthesis: Polynomial Algorithm for Maximal Structure Generation. Computers Chem. Engng. 17, 929-942 (1993)

[12] Friedler, F., Varga, J. B., Fan, L. T.: Decision-Mapping for Design and Synthesis of Chemical Processes: Applications to Reactor-Network Synthesis. In: Biegler, L., Doherty, M. (eds.) AIChE Symposium Series, Vol. 91, pp. 246-250, American Institute of Chemical Engineers, New York (1995)

[13] Friedler, F. Varga, J. B., Feher, E., Fan, L. T.: Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis. In: Floudas, C. A., Pardalos, P. M. (eds.) Global Optimization, Computational Methods and Applications, State of the Art, pp. 609-626, Kluwer Academic Publishers, Dordrecht, Netherlands (1996)

[14] Friedler, F., Fan, L. T., Imreh, B.: Process Network Synthesis: Problem Definition. Networks. 28, 119-124 (1998)

[15] Garcia-Ojeda, J. C., B. Bertok, F. Friedler. "Planning Evacuation Routes with the P-Graph Framework," *Chemical Engineering Transactions*, 29, 1531-1536 (2012)

[16] García-Ojeda, J. C., DeLoach, S. A., Robby, Oyenan, W. H., Valenzuela, J. O-MaSE: A Customizable Approach to Developing Multiagent Development Processes. In: Michael Luck, Lin Padgham (Eds.): Agent-oriented Software Engineering VIII, 8th International Workshop, AOSE 2007, Honolulu, HI, USA, May 14, 2007, Revised Selected Papers, LNCS 4951, 1-15, Springer-Verlag: Berlin (2008)

[17] García-Ojeda, J. C., DeLoach, S. A., Robby: AgentTool III: From Process Definition to Code Generation. In: Proceedings of the 8th international Conference on Autonomous Agents and Multiagent Systems - Volume 2, pp. 1393-1394, Budapest, Hungary (2009)

[18] García-Ojeda, J. C., DeLoach, S. A., Robby: AgentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. In: Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 707-714, Honolulu, Hawaii (2009)

[19] Garro, A., Tundis, A.: A Model-based Method for System Reliability Analysis. In: Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium (TMS/DEVS 2012), Article No 2, Society for Computer Simulation International, San Diego, CA (2012)

[20] Harmon, S. J., DeLoach, S. A., Robby, Caragea, D.: Leveraging Organizational Guidance Policies with Learning to Self-Tune Multiagent Systems. In: Proceedings of the Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, pp. 223-232, IEEE Computer Society, Venice, Italy (2008)

[21] Henderson-Sellers, B., Giorgini, P.: Agent-oriented Methodologies. Idea group Inc. Hershey, PA (2005)

[22] Horváth, A.: Investigation of Failure Systems. Acta Polytechnica Hungarica, 5(2), 127-132 (2008)

[23] Hübner, J. F., Sichman, J. S., Boissier, O.: Developing Organised Multiagent Systems using the MOISE+ Model: Programming Issues at the System and Agent Levels. *Int. J. Agent-Oriented Softw. Eng.* 1(3), 370-395 (2007)

[24] Jin, Y., Levitt, R. E.: The Virtual Design Team: A Computational Model of Project Organizations. Computational & Mathematical Organization Theory. 2(3), 171-196 (1996)

[25] Kauffman, S.: At Home in the Universe: The Search for the Laws of Self-Organization and Complexity. Oxford University Press, Oxford (1995)

[26] Klügl, F.: Measuring Complexity of Multiagent Simulations – An Attempt Using Metrics. In Dastani, M. *et al.* (eds.) LADS 2007, LNAI 5118, pp. 123-138, Springer-Verlag, Berlin Heildeberg (2006)

[27] Kota, R., Gibbins, N. and Jennings, N. R.: A Generic Agent Organisation Framework for Autonomic Systems. In: 1st International ICST Workshop on Agent-based Social Simulation and Autonomic Systems (ABSS 2009), 09-11 Sep, Limassol, Cyprus. pp. 203-219 (2009)

[28] Minoux, M.: Networks Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. Networks, 19, 313-360 (1989)

[29] Nair, R., Tambe, M., Marsella, S.: Team Formation for Reformation. In: Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems, pp. 52-56, AAAI Press, Menlo Park, CA (2002)

[30] Oyenan, W. H., DeLoach, S. A., Singh, S.: An Organizational Design for Adaptive Sensor Networks. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology WI-IAT, pp. 239-242, IEEE computer Society, Toronto, Canada (2010)

[31] Picard, G., Mellouli, S., and Gleizes, M.: Techniques for Multiagent System Reorganization. In: Dikenelli, O. *et al.* (eds.) ESAW 2005, LNAI 3963, pp. 142-152, Springer-Verlag, Berlin Heidelberg (2006)

[32] P-graph – PNS studio, http://www.p-graph.com

[33] Robby, DeLoach S. A., and, Kolesnikov, V. A.: Using Design Metrics for Predicting System Flexibility. In: Baresi, L. *et al.* (eds.) FASE 2006. LNCS, Vol. 3922, pp. 184-198. Springer-Verlag, Berlin Heidelberg (2006)

[34] Serugendo, G. D. M, Gleizes, M. P., Karageorgos, A.: Self-Organisation and Emergence in Mas: An Overview. Informatica 30(1), 45-54 (2006)

[35] Sims, M., Corkill, D., Lesser, V.: Automated Organization Design for Multiagent Systems. Auton. Agents and Multiagent Syst 16(2), 151-185 (2008)

[36] Vazquez-Salceda, J., Dignum, F.: Modelling Electronic Organizations. In Mark, V. *et al.* (eds.) Multiagent Systems and Applications III, LNAI 2691, pp. 584-593 Springer-Verlag, Berlin Heidelberg (2003)

[37] Zhong, C., DeLoach, S. A.: An Investigation of Reorganization Algorithms. In: 2006 International Conference on Artificial, pp. 514-517, CSREA Press (2006)

[38] Zhong, C.: Integrating Humans into and with Computing Systems. Ph.D. Dissertation, Kansas State University (2010)