On Modeling Building-Evacuation-Route Planning and Organization-based Multiagent Systems by Resorting to the P-graph Framework

DOI: 10.18136/PE.2016.630

Doktori (PhD) értekezés

Juan Carlos García Ojeda

Supervisor: Dr. Friedler Ferenc

Co-Supervisor: Dr. Bertok Botond

University of Pannonia

Department of Information Technology

Information Science and Technology PhD School

Veszprém, Hungary

2015

ON MODELING BUILDING-EVACUATION-ROUTE PLANNING AND ORGANIZATION-BASED MULTIAGENT SYSTEMS BY RESORTING TO THE P-GRAPH FRAMEWORK

Dissertation for obtaining a PhD degree

Written by: Juan Carlos García Ojeda

Written in the Information Science and Technology Doctoral School
of the University of Pannonia

Supervisor: Dr. Friedler Ferenc

I propose for acceptance (yes / no)

……………………….
(signature)

The candidate has achieved …........ % at the comprehensive exam,

I propose the dissertation for acceptance as the reviewer:

Name of reviewer: …...................... …................. yes / no

……………………….
(signature)

Name of reviewer: …...................... …................. yes / no

……………………….
(signature)

The candidate has achieved …........ % at the public discussion.

Veszprém/Keszthely,

....…………………………
Chairman of the Committee

Labelling of the PhD diploma ….................................

.…..…………………………
President of the UCDH

Table of Contents

List of Figures

List of Tables

Nomenclature

The most important notations and acronyms used throughout this Dissertation are listed below.

Mathematical Notation

$achieves$	defines the extent of achievement of a goal by a role
a_{ji}	the difference between the production and consumption rate of the j^{th} element in M_j by the i^{th} element in O_i
$\|A\|$	cardinality, or size, of set A
A_{OMACS}	the set of agents, which can be either human or artificial (hardware or software) entities
$capable$	defines the set of capabilities required to play a role
$cost$	defines the cost of an agent
C_{OMACS}	the set of capabilities, which define the percepts/actions the agents possess at their disposal. Capabilities can be soft (i.e., algorithms or plans) or hard (i.e., hardware related actions).
c_{P_j}	the cost of the j^{th} element in P_j
cp_{O_i}	the proportional cost of the i^{th} element in O_i
ϕ_{OMACS}	the subset of all the potential assignments of agents to play roles to achieve goals
$\phi_{assignments}$	contains the best possible assignments $\langle a_i, r_k, g_j \rangle$ for the given sets A_{OMACS}, R_{OMACS}, and G_{OMACS}
G_{OMACS}	the set of goals of the organization, i.e., overall functions of the organization
∞	infinite
k	number of iterations
L	large number, i.e., $L = \infty$
L_{P_j}	the lower bound of the j^{th} element in P_j

L_{O_i}	the lower bound of the i^{th} element in O_i
L_{R_j}	the lower bound of the j^{th} element in R_j
M	the set of entities
n	total number of states in S
oaf	defines the quality of a proposed set of assignments, i.e., oaf computes the goodness of the organization based on Φ_{OMACS}
$\emptyset$	empty set
O	the set of activities
O_i	the i^{th} element in O
O_{OMACS}	the multiagent system's organization
$possesses$	defines the quality of an agent´s capability
$potential$	defines how well an agent can play a role to achieve a goal
$\wp$	denotes power set
P	the set of products
p_{ij}	the *one-step transition probability* is the probability of transitioning from one state, i.e., i, to another, i.e., j, in a single step
P_j	the j^{th} element in P
P_{Markov}	absorbing markov chain
P_{OMACS}	the set of rules that describe how $\mathrm{O_{OMACS}}$ may or may not behave in particular situations
$requires$	a function that assumes a role in $\mathrm{R_{OMACS}}$, thereby yielding a set of capabilities required to play that role
$\mathbb{R}$	set of real numbers
R	the set of initially available resources
R_j	the j^{th} element in R
R_{OMACS}	the set of roles, i.e., positions within an organization, whose behavior is expected to achieve a particular goal or set of goals
U_{O_i}	the upper bound of the i^{th} element in O_i

U_{P_j}	the upper bound of the j^{th} element in P_j
U_{R_j}	the upper bound of the j^{th} element in R_j
Σ_{OMACS}	the environment where agents can perform their actions upon it.
S	state space of P_{Markov}
S_F	the set of pairs representing the transition from state i to state $failure$ with probability $p_{ifailure}$
S_S	the set of pairs representing the transition from state i to state $success$ with probability $p_{isuccess}$
S_T	the set of pairs representing the transition from state i to state j with probability p_{ij}
$x_n^{(k)}$	probability vector of P_{Markov}
x_{O_i}	continuous variable expressing the size of capacity of the i^{th} element in O_i
y_{O_i}	binary variable, i.e., $y_{O_i} \in \{0,1\}$, expressing the absence (0) or existence (1) of the i^{th} element in O_i
z	the objective value

Acronyms

ABB	optimal solution structure generator algorithm
BEP	building evacuation problem
$BEPtoPNS_T$	algorithm for transforming a building evacuation problem (BEP) to the corresponding time-expanded process-network synthesis problem (PNS_T)
CDF	cumulative density function
$CRST$	cooperative robotic search team
FR	failure rate
LP	linear programming
$MILP$	mixed integer linear programming

MSG	maximum structure generator algorithm
PDF	probability distribution function
PNS	process network synthesis problem
PNS_T	time-expanded process-network synthesis problem
$OMACS$	organization model for adaptive computational systems framework
$O - MaSE$	organization-based multi-agent software engineering methodology framework
SSG	structure generator algorithm
UML	unified modeling language

Abstract

This work is motivated by our deep conviction about the role of optimization models in real world problems. To this extent, this dissertation presents the work carried out in two seemingly unrelated domains: building-evacuation-route planning, and modeling organization-based multiagent systems. Both domains are seen from a wider perspective as instances of optimization models, where the common outcome is concerned with the minimization or maximization of a certain function, possibly under constraints.

With regards to the building-evacuation-route planning problem, a method and software for optimal building-evacuation-route planning are proposed in terms of identifying evacuation routes and scheduling of evacuees on each route. First, the building-evacuation routes are represented by a P-graph, which gives rise to a time-expanded process-network synthesis (PNS_T) problem that can be algorithmically solved according to the P-graph framework; each location and passage in the building are given by a set of attributes to be taken into account in the evacuation-route planning. The evacuation time is calculated as a minimum cost of the corresponding PNS_T. In addition to the globally optimal solution, the P-graph framework provides the n-best sub-optimal solutions. The validity of the proposed method is illustrated by several examples.

With respect to the modeling of organization-based multiagent systems problem, at the outset, the design of organization-based multiagent systems is proposed according to the framework of Organization Model for Adaptive Complex Systems (OMACS). Subsequently, this design model is transformed into a process-network model, i.e., P-graph. 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 combinatorially feasible structures, i.e., OMACS-based design configurations, capable of yielding the specified products from the specified raw material; (ii) every combinatorially feasible structure of 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.

Resumen

Este trabajo está motivado por nuestra profunda convicción sobre el papel de los modelos de optimización en los problemas del mundo real. En este sentido, esta disertación presenta la labor llevada a cabo en dos dominios aparentemente no relacionados: planeación de rutas de evacuación en edificios, y modelados de sistemas multiagente basados en organizaciones. Ambos dominios se pueden ver desde una perspectiva más amplia como ejemplos de modelos de optimización, en el que el resultado común tiene que ver con la minimización o maximización de una función determinada, posiblemente bajo restricciones.

En cuanto al problema de planeación de rutas de evacuación en edificios, se propone un método y un software para la planeación de rutas de evacuación en edificios en términos de identificar las rutas de evacuación y la programación de los evacuados en cada ruta. En primer lugar, las rutas de evacuación del edificio se representan mediante P-graph, lo que da lugar a un problema de síntesis de redes de procesos de tiempo extendido (PNS_T) que se puede resolver algorítmicamente de acuerdo con P-graph; cada lugar y espacio en el edificio son definidos por un conjunto de atributos que deben tenerse en cuenta en la planeación de de las rutas de evacuación. El tiempo de evacuación se calcula como un coste mínimo de correspondiente PNS_T. Además de la solución óptima general, P-graph proporciona las n-mejores soluciones sub-óptimas. La validez del método propuesto se ilustra con varios ejemplos.

Con respecto al problema de modelado de sistemas multiagente basados en organizaciones, en principio, se propone el diseño de sistemas multiagente basados en organizaciones de acuerdo con modelo Organization Model for Adaptive Complex Systems ($OMACS$). Posteriormente, este modelo de diseño se transforma en un modelo de redes de procesos, es decir, P-graph. Finalmente, el modelo de redes de procesos resultante en conjunción con la metodología P-graph de lugar a: (i) la estructura máxima de la red de proceso, que comprende todas las estructuras combinatoria viables, es decir, configuraciones de diseño basados en $OMACS$, capaces de obtener los productos especificados a partir de la materia prima especificada; (ii) toda estructura combinatoria posible del proceso de interés; y (iii) la estructura óptima de la red, es decir, la configuración de diseño óptimo basado en $OMACS$. Por último, a la luz del principio de un paradigma de modelado-transformación-evaluación, una evaluación se hace de la viabilidad, así como la flexibilidad y el coste de la configuración de diseño óptimo obtenido basado en $OMACS$.

Összefoglaló

Ezt a munkát az a mély meggyőződés motiválja, hogy az optimalizálási modellek elősegítik gyakorlatban felmerülő problémák megoldását. Ennek érdekében ez az értekezés két, egymástól látszólag független területen – jelesül épület-kiürítési útvonalak tervezése, illetve szervezeti alapú, multiágens rendszerek modellezése terén - elvégzett munkát mutat be. Mindkét területet tágabb perspektívából optimalizálási modellek eseteiként fogjuk fel, ahol a közös eredmény egy bizonyos függvény minimalizálásával vagy maximalizálásával jön létre, esetleg korlátok között.

Ami az épület-kiürítési útvonalak tervezésének problémáját illeti, kidolgoztunk egy módszert és egy szoftvert optimális épület-kiürítési útvonalak tervezésére: az evakuálási útvonalak azonosítása és az egyes útvonalakon evakuálandók ütemezése tekintetében. Először is, az épület-kiürítési útvonalakat egy P-gráf representálja: ez egy időben elnyújtott folyamat-hálózati szintézis (PNS_T) problémáját veti fel, amely algoritmikusan megoldható a P-gráf keret szerint; az épület minden egyes helyét és a folyosóját egy sor jellemző határozza meg, amelyeket figyelembe kell venni a kiürítési útvonal tervezésében. A kiürítési időt a megfelelő PNS_T minimális költségeként kalkuláljuk. A globálisan optimális megoldás mellett a P-gráf keret megadja az n-edik legjobb szuboptimális megoldásokat is. A módszer érvényességét több példával szemléltetjük.

Ami a szervezeti alapú multiágens rendszerek modellezésének problémáját illeti, kezdetben szervezeti alapú multiágens rendszerek megtervezését javasoljuk a Komplex Adaptív Rendszerek Szervezeti Modelljének ($OMACS$) kerete szerint. Ezt követően ezt a tervezési modellt átalakítjuk folyamat-hálózati modellé, azaz a P-gráffá. Majd a kapott folyamat-hálózati modell a P-gráf alapú metodológiával együtt létrehozza: (i) a folyamat-hálózat maximális struktúráját, amely magába foglalja az összes kombinatorikusan megvalósítható struktúrát, azaz $OMACS$ alapú tervezési konfigurációt, amely képes produkálni a meghatározott termékeket a meghatározott alapanyagból; (ii) az érintett folyamat valamennyi kombinatorikusan megvalósítható struktúráját; és (iii) a hálózat optimális szerkezetét, azaz az optimális $OMACS$-

alapú tervezési konfigurációt. Végül egy modellezés-átalakítási-értékelési paradigma tételének fényében, felmérjük a kapott optimális $OMACS$ alapú tervezési konfiguráció megvalósíthatóságát, valamint rugalmasságát és költségét.

Acknowledgements

The author wishes to express his most profound appreciation to Dr. Ferenc Friedler of Faculty of Information Technology at Pázmány Péter Catholic University and Dr. L.T. Fan (†) of Department of Chemical Engineering at Kansas State University for their excellent guidance, constant encouragement an invaluable advice for finishing this work. Appreciation is extended to Dr. Botond Bertok of Department of Computer Science and Systems Technology at Pannonia University for his advice and guidance. Special gratitude is also extended to Dr. Andres Argoti for his advice, assistance, and friendship. Special thanks go to Dr. Itsvan Heckl, Mate Hegyhati, Orsolya Kristof, Zita Vereskuti, Freddy Mendez Ortiz, and Dr. Laszlo Palotas. The financial support provided by the Department of Computer Science and Systems Technology at Pannonia University is gratefully acknowledged.

Dedication

To my parents, Isidro Elías (†) and Evila, my wife Lina Marcela, and my son Santiago.

Chapter 1. Introduction

This section presents the work done in two seemingly unrelated research domains: building-evacuation-route planning, and modeling organization-based multiagent systems. However, my interest in these research domains derives from the same source; that is, our deep conviction about the role of optimization models. Both problems can be seen from a wider perspective as instances of optimization models where the common outcome is concerned with the minimization or maximization of a certain function, possibly under constrains.

1.1 Building-Evacuation-Route Planning

Route Evacuation Planning is the science of ensuring the safest and most efficient evacuation time of all expected residents of a building, city or region, or transportation carriers (e.g., train, ship, and airplane) from a treat or actual occurrence of a hazard (e.g., natural disasters, traffic, industrial, or nuclear accidents, fire, viral outbreak, etc.) [47]. In any scenario (i.e., building, city or region, or transportation carriers), a proper planning may imply the evaluation of a countless number of evacuation routes which is considerably challenging because of the combinatorial nature of the problem [97]. Naturally, towards this end, it is highly desirable or even essential, to have access to optimization software. Such software should be able not only to generate an optimal evacuation plan, but also to yield and evaluate every feasible evacuation plan [17,22], whenever computationally possible, due to its complexity [100].

In the particular case of building evacuation, the occupants' evacuation is one of the most important concepts of the buildings safety. For this reason, buildings are safe if they are built according to local building authority regulations and codes of practice. However, it is not always necessary to evacuate a building during an emergency. For instance, a power outage does not necessarily call for an evacuation [9]. Current research efforts fall into six categories[1] [11,71]: level of service, mathematical models, heuristics methods, stochastic models, simulation tools, and multiagent systems.

In this dissertation, we focus on mathematical models for generating optimal evacuation plans which minimize the total evacuation time. Mathematical models adopt flow networks algorithms to evaluate the routes (e.g., minimum cost flow, maximum flow, quickest path, etc.). Mathematical models rely on the category of level of service research for characterizing the walking speed and spacing between evacuees based upon the density of evacuees using a pathway or corridor [61,81,84,85,86,87,90,91,106].

Even though these evacuations planning algorithms generate optimal plans, they are computationally expensive with respect to the resources they can use (e.g., memory and processing time) [47,107]. For example, Francis, in [28,29], proposes the application of mathematical optimization for building evacuation by adopting Brown´s algorithm [7]. Then, Berlin points out the use of flow networks in building evacuation [4] followed by Francis *et al*'s works [13,60]. These works are later on extended to consider problems where flow networks are constrained by their capacities and solved by adopting greedy and polynomial algorithms [15,53,54]. Other works focus on formulating the building evacuation as a multi-objective problem [44,45,66,112].

To overcome the computational cost of computing building-evacuation-routes-plans by resorting to mathematical models, heuristic models are proposed [74][2]. Also, stochastic models are adopted to capture the overall egress process more realistically, despite the fact their resolution is more laborious [73,102,103,104]. In recent years, simulation methods have gained adepts. Simulation methods model and emulate traffic flow and assume that the behavior of individuals is under the influence of other. Three approaches have been adopted for simulating traffic flow [47]: probabilistic models [22,73], cellular automata

[1] In most of the cases, these categories take advantage of the advances in the Geographical Information Systems field for accessing data or drawing graphical location-based information.

[2] Although the do not always generate the optimal solution.

[3,5,19,64,80], and multiagent systems [10,62,96,110]. In [67], a list of simulation models and software packages for simulating pedestrian motion can be found.

In this dissertation, we are to examine and propose a MILP model based upon the traditional discrete time dynamic network flow model [47]. This model will be explained next.

1.1.1 Discrete Time Dynamic Network Flow Model

A discrete time dynamic network flow model is a discrete time expansion of a static network flow problem, where the flow is distributed over a set of predetermined time periods $t = 0,1, \dots, T$ [47].

In [47] a definition of dynamic network flow model is introduced. Let $G = (N, A)$ be a directed network with N the set of nodes and A the set of arcs. Travel time λ_{ij} is given for each arc $(i, j) \in A$; where λ_{ij} is assumed to be constant. The time expansion of G over a time horizon T defines the dynamic network $G_T = (N_T, A_T)$ associated with G where

$$N_T = \{i(t) | i \in N; t = 0,1, \dots, T\}$$

and A_T consist of movement arcs A_M, where

$$A_M = \{(i(t), j(t')) | (i, j) \in A; t' = t + \lambda_{ij} \leq T; t = 0,1, \dots T\}$$

and the set of holdover arcs A_H

$$A_H = \{(i(t), i(t + 1)) | i \in N; t = 0,1, \dots T - 1\}$$

i.e.,

$$A_T = A_M \cup A_H$$

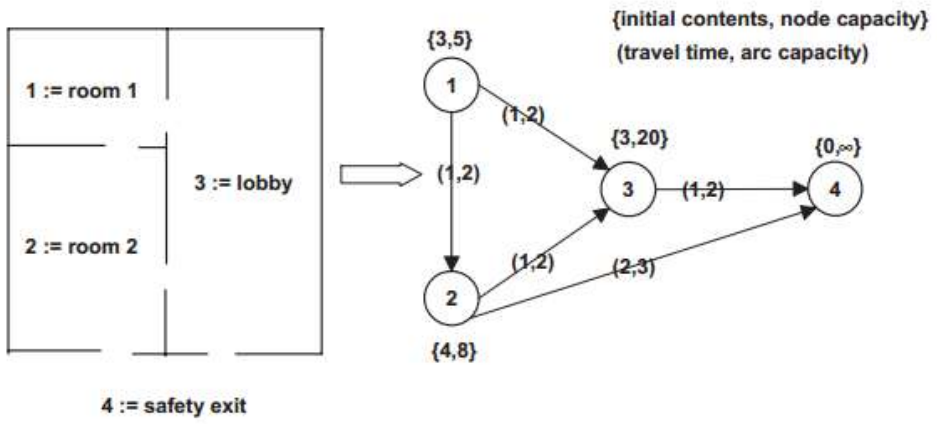


Figure 1. Static network G of a simple building layout (taken from [47]).

To construct the dynamic network, G_T defined above, the following assumptions have been made. First, the time period t is dependent of θ (the basic time unit) in which travel times are measured. For instance, if we choose $10\ s$ on the length of the basic unit, i.e., $\theta = 10$, then specifying three times period, i.e., $T = 3$, for traversing an arc means an evacuee needs thirty seconds to do so. It can be noticed that, the smaller θ the more accurately the model represents the actual flow's evolution[3].

Since the dynamic network has $(T + 1)$ copies of each source-node and each sink-node, the dynamic network will have multiple sources and sinks. Therefore, in order to reduce the size of the dynamic network, a super-source s and a super-sink d are introduced to create a single source/sink network (see Figure 2). In evacuation problems, d can be interpreted as a common safety area; and, s the place where all evacuees are initially located. For every source-node, a holdover arc is created. Holdover arcs from s to source-nodes have zero travel time and capacities are equal to initial occupancies. In the maximum dynamic flow problem, s is connected to all time copies of the source-nodes (e.g., node 1 in Figure 3). On the other hand, generally, all copies of every sink-node are connected to d; hence, there is no holdover arc for sink-nodes. All connections to d have zero travel time and infinite flow capacities. Nevertheless, it can be noted that, dynamic network flow

[3] However, choosing θ too small will result in undesirable size of the network. Hence, the choice of θ is a compromise between model realism and model complexity [47].

problems can always be solved as static flow problems in the expanded network. Also, the equivalent static problem does not require keeping arc capacities and travel times fixed over time, as assumed before, but these assumptions are essential for building efficient algorithms to solve the problem [47]. The upper bound for the number of nodes and arcs in discrete time dynamic network can be stated as follows. If $n = |N|$ and $m = |A|$ then $n(T+1)$ and $(n+m)T + m - \sum_{(i,j)\in A} \lambda_{ij}$ are the upper bound for the number of nodes and arcs in G_T without considering super-source and super-sink, respectively [47]. Since arc in the path from s to any sink-node at time t are greater than T, the size of the time-expanded network can be reduced by eliminating inessential arcs including their corresponding nodes (see Figure 3).

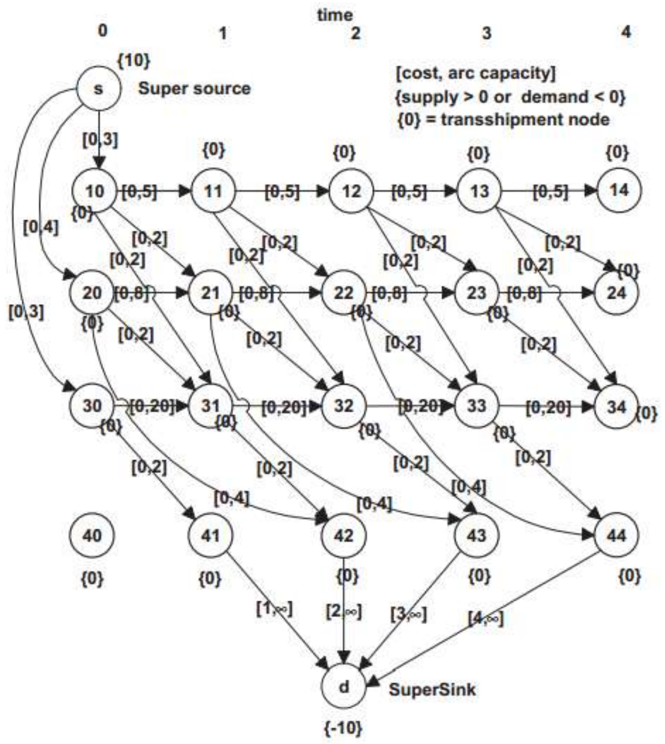


Figure 2. Dynamic Network G_T of the Static Network G of Figure 1, with $T = 4$ (taken from [47]).

In the dynamic network flow models, $x_{ij}(t)$ denotes the flow (e.g., the number of evacuees moving at time t) that leaves node i at time t and reaches node j at time $t + \lambda_{ij}$. Flows from node i at time t to the same node with travel time λ_{ii}=1 represents the number of evacuees who prefer to stay in the building component represent by node i at time t for at least one unit of time. This flow is denoted by $y_i(t+1)$, i.e., $y_i(t+1) = x_{i(t),i(t+1)}$.

The capacity of movement arcs $\left(i(t), j\left(t+\lambda_{ij}\right)\right) \in A_M$ is denoted by $b_{ij}(t)$ where, without loss of generality, we can assume that

$$b_{ij}(t) = min\left\{b_{ij}(t') | t' = t, t+1, \dots, t+\lambda_{ij}\right\}$$

The capacity of a holdover arc $\left(i(t), j(t+1)\right) \in A_H$ is determined by the node capacity $a_i(t)$, and represents how many evacuees can stay in the node i at a given time t. With $\theta(X,Y)$ as the general objective and with q_i as the initial number of evacuees in any node $i \in N$, gives rise to the discrete-time dynamic network flow model for evacuation process.

$$min/\max \quad \sum_{t=0}^{T} \theta(X,Y) \tag{1.1}$$

subject to

$$y_i(t+1) - y_i(t) = \sum_{k \in pred(i)} x_{ki(t-\lambda_{ki})} - \sum_{j \in succ(i)} x_{ij(t)} \qquad t = 0, \dots, T; \forall i \in N \tag{1.2}$$

$$y_i(0) = q_i \qquad \forall i \in N \tag{1.3}$$

$$0 \le y_i(t) \le a_i(t) \qquad t = 1, \dots, T-1; \ \forall i \in N \tag{1.4}$$

$$0 \le x_{ij}(t) \le b_{ij}(t) \qquad t = 0, \dots, T-\lambda_{ij}; \ \forall i \in N \tag{1.5}$$

where

$$pred(i) = \{j|(j,i) \in A\}; succ(i)\{j|(i,j) \in A\}$$

are the nodes which are predecessors and successors of node i, respectively.

In order to measure the time when evacuees reach their final destinations, so-called turnstile cost [13,48] is defined on each arc (see arcs (41,d), (42,d), (43,d), and (44,d) in Figure 3) as follows; if D is the set of sink nodes of the static network G and d is the super sink node of the associated dynamic network G_T, the (turnstile) cost of any arc $\left(i(t), j\left(t' = t + \lambda_{ij}\right)\right) \in A_M$ is defined different from 0 iff $i \in D$ and $j(t') = d$. In this case [47], $c(i(t), d) = t$.

Let $S \subset N$ denote the set of source-nodes of the static network G. Using the previous definition of turnstile cost, the objective function $\theta(X,Y)$ to model the average time required by an evacuee to leave the network can be stated as follows [47].

$$\theta(X,Y) = \frac{\sum_{t=0}^{T} \sum_{i \in D} t x_{id}(t)}{\sum_{i \in S} q_i} \tag{1.6}$$

Since the denominator is constant and θ depends only on the flow variables, one just need to define the objective function θ as

$$\theta(X,Y) = \theta(X) = \sum_{t=0}^{T} \sum_{i \in D} t x_{id}(t) \tag{1.7}$$

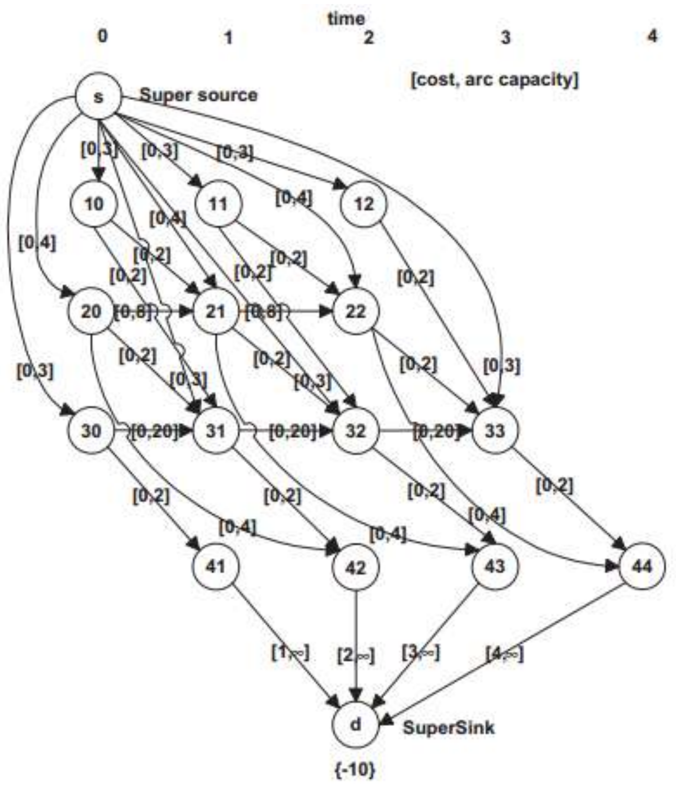


Figure 3. Dynamic network G_T of the static network G of Figure 1, with $T = 4$, without initial contents, and by deleting inessential arcs (taken from [47]).

Finally, the movement of initial occupancies are modeled by using flow from s to each source-node. Assuming constant capacity (i.e., $b_{ij}(t) = b_{ij}, \forall (i,j) \in A$ and $a_i(t) = a_i,\ \forall i \in N; t = 0, \dots, T)$ of each node and constant travel time between them gives rise to the evacuation model (LP) that minimizes the average evacuation time.

$$min \sum_{t=0}^{T} \sum_{i \in D} t x_{id}(t) \tag{1.8}$$

subject to

$$x_{si}(0) = q_i \qquad \forall i \in S \tag{1.9}$$

$$\sum_{t=0}^{T} \sum_{i \in D} t x_{id}(t) = \sum_{j \in S} q_j, \tag{1.10}$$

$$y_i(t+1) - y_i(t) = \sum_{k \in pred(i)} x_{ki(t-\lambda_{ki})} - \sum_{j \in succ(i)} x_{ij(t)} \qquad t = 0, \dots, T; \forall i \in N$$

$$y_i(0) = 0 \qquad \forall i \in N \tag{1.11}$$

$$y_i(t) = 0 \qquad \forall i \in D; t = 0, \dots, T \tag{1.12}$$

$$0 \leq y_{ij}(t) \leq a_i(t) \qquad t = 1, \dots, T; \; i \in N - D \tag{1.13}$$

$$0 \leq x_{ij}(t) \leq b_{ij}(t) \qquad t = 0, \dots, T - \lambda_{ij}; \; \forall (i,j) \in A \tag{1.14}$$

As result, a time-expanded network, as defined in the first definition introduced in this section, can be evaluated as a static network and then solved by applying any minimum cost static network flow algorithm to obtain the solution [1,47].

1.2 Organization-Based Multiagent Systems

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 [18]. Trends in the field of autonomous agents and multiagent systems suggest that the explicit design and use of organization-based multiagent systems [76], which allow heterogeneous agents (either human or artificial entities) rely on well-defined roles to accomplish either individual or system level goals [21,114], is a promising approach to these new requirements [76].

When focusing on system's goals, an organization of agents allows its members, i.e., individual agents, to work together to perform the tasks for which they are best suited. When emphasizing an individual agent's goal, an organization provides the infrastructure that allows agents to find and carry out collaborative tasks with other entities to the mutual benefit. In situations where the nature of the environment makes the organization susceptible to individual failures, these failures can significantly reduce the ability of the organization to accomplish its goals.

In the literature a set of methodologies [52], a selection of design processes [16], and a collection of frameworks [18,20,24,26,55,65,99] are available to provide the basis for constructing sophisticated autonomous multi-agent 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 multi-agent organizations [56,63,88,95].

The above-mentioned methodologies and frameworks, however, do not offer techniques for identifying the number of feasible configurations of agents that can be synthesized, or designed, from a set of heterogeneous agents. This is an important issue in designing a multiagent system because of the nature of the environments where it operates (dynamic, continuous, and partially accessible) [81]. 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 [65,81].

In this dissertation, we are to examine and propose a couple of organization-based multiagent systems assessment models based upon the framework OMACS [18]. This framework will be explained next.

1.2.1 Overview of 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 multi-agent systems and their relationship as a tuple $O_{OMACS} = \langle G_{OMACS},\ R_{OMACS},\ A_{OMACS},\ C_{OMACS},\ \Phi_{OMACS},\ P_{OMACS},\ \Sigma,\ oaf,\ achieves,\ capable, requires,\ possesses,\ potential\rangle$, and it is also represented via an UML[4]-based organizational meta-model (see Figure 4) [18]. These are briefly described in what follows.

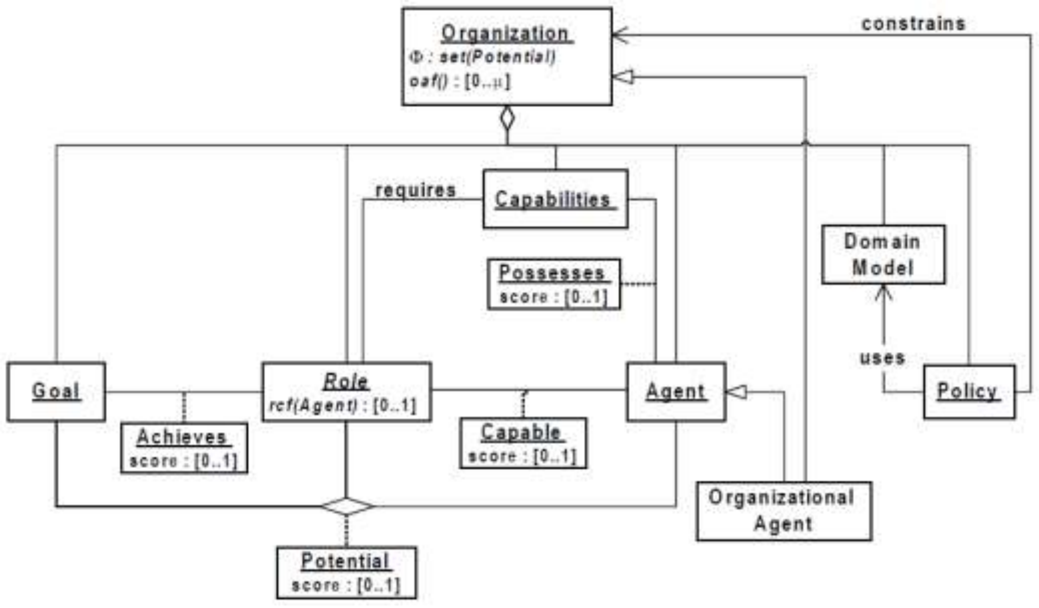


Figure 4. OMACS Meta-model.

[4] 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.

In addition, OMACS defines a set of functions – $achieves$, $requires$, $possesses$, $capable$, $potential$, oaf, and Φ_{OMACS} – 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, thereby yielding a set of capabilities required to play that role ($requires$, $R_{OMACS} \rightarrow \wp(C_{OMACS})$, defines the set of capabilities required to play a role[5]); $capable$, a function whose inputs are an agent in A_{OMACS} and a role in R_{OMACS} and generates an output, which is a positive real number greater than or equal to 0 and less than or equal to 1 ($capable$, $A_{OMACS}\,x\,R_{OMACS} \rightarrow$ [0,1], defines how well an agent can play a role), thus giving rise to

5 $\wp$ denotes power set.

$$capable(a,r) = \begin{cases} 0 & if \prod_{c \in requires(r)} possesses(a,c) = 0 \\ \frac{\sum_{c \in requires(r)} possesses(a,c)}{|requires(r)|} & elsewhere; \end{cases} \quad (1.15)$$

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

$$potential(a,r,g) = achieves(r,g) * capable(a,r); \quad (1.16)$$

($potential$, $A_{OMACS}\ x\ R_{OMACS}\ x\ G_{OMACS} \rightarrow [0,1]$, defines how well an agent can play a role to achieve a goal), and assignment set, Φ, the set of agent-role-goal tuples $\langle a,r,g \rangle$, 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 $[0,\infty]$ (oaf, $\wp(\Phi_{OMACS}) \rightarrow [0,\infty]$, defines the quality of a proposed set of assignments, i.e., oaf computes the goodness of the organization based on Φ_{OMACS}), thus resulting in

$$oaf = \sum_{<a,r,g> \in \Phi_{OMACS}} potential(a,r,g). \quad (1.17)$$

1.3 Objectives

The work presented here in this dissertation aims at mathematical modeling of two apparently unrelated research domains: building-evacuation-route planning, and modeling organization-based multiagent systems. Specific objectives of this work are as follows:

a) For the Building Evacuation Route Planning Problem

(i) To transform it into a P-graph model taking into account the temporal dimension inherent to the building evacuation problem in terms of the evacuation time, specifically, its upper bound T.

(ii) To calculate the evacuation time as a minimum cost of the resultant MILP model

(iii) To validate the results of the resultant MILP model in light of the available experimental data taken from the literature.

(iv) To evaluate the existence of n-best sub-optimal solutions.

b) For the Modeling of Organization-based Multiagent System Desing Problem

(i) To transform design of organization-based multiagent systems, according to the framework OMACS, into a P-graph model

(ii) To solve algorithmically the resultant MILP model

(iii) To validate the results of the resultant MILP model in light of simulated data.

(iv) To evaluate the existence of n-best sub-optimal solutions.

Besides the current chapter, this dissertation contains four additional chapters, i.e., Chapters 2 through 5.

Chapter 2 presents the analysis, modeling, and evaluation of building-evacuation-route planning. Chapter 3 focuses on the analysis, modeling, and simulation of organization-based multiagent systems. Chapter 4 draws the mayor conclusions and recommendation for possible extensions are proposed. Finally, the major outcomes of this dissertation are presented in Chapter 5.

Chapter 2. Building-Evacuation-Route Planning: Research Results

2.1 Background

The aim of any building evacuation plan is to ensure the safest and fastest movement of people away from any threat (e.g., bomb threat and taking of hostages) or the occurrence of a hazard (e.g., industrial or nuclear accidents, natural disasters, fire, and viral outbreak) [105]. Nevertheless, buildings are increasingly built taller and more complex, thus rendering it difficult to establish a rapid evacuation plan [92].

In any emergency scenario, determining an optimal or near optimal evacuation plan, in terms of the egress time, entails the evaluation of a myriad of evacuation routes, which is highly convoluted because of the combinatorial nature of the problem [17,47,58,100]. Naturally, towards this end, it is highly desirable or even essential, to have access to optimization software. Such software should be able not only to generate an optimal evacuation plan, but also to yield and evaluate every feasible evacuation plan [17,22], whenever computationally possible, due to its complexity [100]. Egress models such as EVACNET4, WAYOUT, and PathFinder which employ optimization software generate at most one globally optimal solution for showing congestion areas, queuing, or bottlenecks [70,72].

Following, an algorithmic method for optimizing a building evacuation plan, in terms of the egress time, supported by software tools at each step is presented. This method resorts to the graph-theoretic approach based on the P-graph framework. The method is demonstrated by applying it to the evacuation of different building configurations (i.e., one-story building, a two-story, a three-story, and a ten-story building).

2.2 Problem Definition

Let $G = (N, A)$ be a directed graph with N the set of nodes and A the set of arcs. For an evacuation plan, the potential locations of evacuates and other areas, e.g., rooms, corridors, safe areas, stairs, or intersections, on a building-floor map are represented by nodes $n \in N$, and the potential movements between the locations, through k, e.g., passages, gates, or doorways, or edges, by arcs $(i, j, k) \in A$ and $i, j \in N$; see Figure 5[6]. We are to minimize the time of a building evacuation plan consisting of a set of evacuation routes and a scheduling of evacuees on each route.

The evacuation plan should satisfy the constraints imposed by the building itself [47]. Specifically, each location n has a limited capacity expressed by non-negative integer cap_n, which is the number of individuals that can be accommodated at this location. The initial occupancy is also assigned to each location n by non-negative integer ic_n, which is the number of individuals at any given location in the event of an emergency. Moreover, the maximum flow rate of passage (i, j, k) is defined by positive integer $cap_{(i,j,k)}$. The flow rate is the maximum number of individuals that can travel through it simultaneously. Passages may act as bottleneck points in the floor-map. Finally, each passage (i, j, k) is constrained by non-negative travel time $\lambda_{(i,j,k)}$. Travel time is a measure of time required by an individual to go through the entire length of a passage. Additionally, it is noteworthy to mention that it is up to the expert or group of experts the process of specifying each of the constraints, mentioned previously, for the building of interest [13,14,47,49,50,59,70,72,94].

[6] The graph-based notation has been slightly modified to introduce variable k from its original [47].

The graph-theoretic approach based on P-graphs (process graphs) has been conceived for optimally synthesizing a process network presumably operating under steady-state, or stationary, conditions; naturally, no temporal dimension is involved [31,33,34,35]. Figure 6 shows an approximation of a PNS problem (represented via p-graph) of the building evacuation problem introduced in Figure 5. For a building evacuation problem the locations of evacuees including safe areas (e.g., rooms, corridors, safe areas, stairs, or intersections) on the building-floor map are represented by entities $m \in M$, the initial location of evacuees are represented by raw materials $r \in R$; and, the potential movements between the locations (through, e.g., passages, gates, or doorways, and edges) by activities $o \in O$ (see Figure 6) [42,43]. It is noteworthy to mention that this representation raises some issues regarding the p-graph model and the building evacuation problem. First, the p-graph model violates axiom (S2). That is, B and C (both raw materials) are produced by operating units (1,2,1) and (1,2,2), respectively. Second, this model does not capture the temporal dimension of the problem in terms of the egress time of the individuals inside the building at the onset of an emergency. Therefore, a new approach is required.

To deploy this approach for the problem of interest entails an appropriate adaptation of the approach, PNS problem, to take into account the temporal dimension inherent to the problem in terms of the evacuation time, specifically its upper bound T [47,100]. This have given rise to the development of the time-expanded process-network synthesis, PNS_T, proposed in the current dissertation [41,42,43].

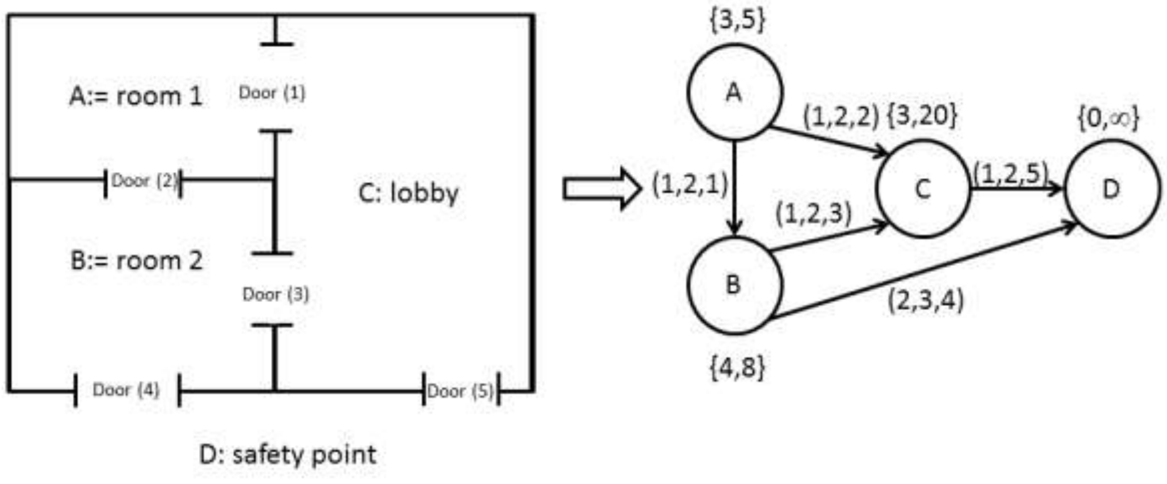


Figure 5. Conventional adopted graph-based notation for representing building-floor maps [47]: {initial contents, node capacity}; (travel time, arc capacity, arc id).

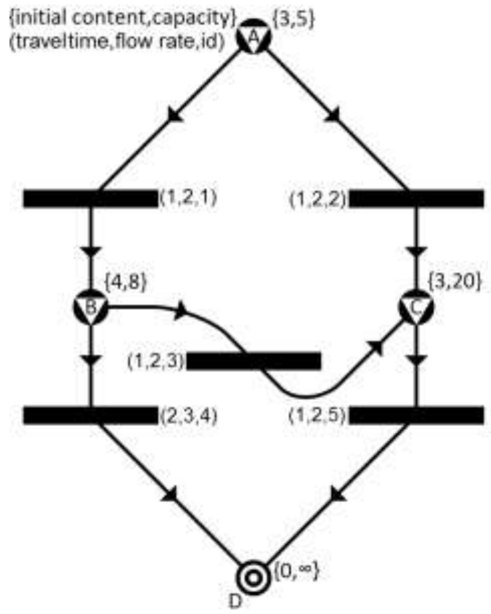


Figure 6. P-graph representation of the building floor map introduced in Figure 5.

2.3 Methodology

2.3.1 P-graph-based approach

This approach is rooted in the two cornerstones; one is the P-graph representation of a process network of interest, and the other is a set of five axioms for solution structures, i.e., combinatorial feasible networks. These two cornerstones render it possible to fashion the three mathematically rigorous algorithms, including algorithm MSG (maximum-structure generation), algorithm SSG (solution-structure generation), and algorithm ABB (accelerated branch-and-bound). These three algorithms are capable of not only generating exhaustively and exclusively solutions structures but also of identifying exactly the globally optimal structure, i.e., network, near optimal structures in ranked order ([31,33,34,35]; also see Appendixes A and B).

2.3.2 Time-expanded process-network synthesis, PNS_T

Given an upper bound $T$of the evacuation time and set M of entities, i.e., locations and the state of locations at time t, $0\leq t < T$, a PNS_T problem is given by triplet (P, R, O). In this triplet: set $P \subseteq M$ contains the final target to be reached, i.e., common safety point [25]; set $R \subseteq M$ contains the initially available resources, i.e., locations of individuals; and set $O \subseteq \wp(M) \times \wp(M)$ comprises the candidates activities for forming a network to reach each of the final targets by moving the total amount of available resources, i.e., the potential movements of evacuees between the locations. Each activity o is defined by a pair of its preconditions and outcomes, i.e., for each $o \in O : o = (\alpha, \beta)$, where $\alpha, \beta \subseteq M.$ A precondition can be the availability of a resource or an outcome of another activity.

In any evacuation scenario, the initial locations of individuals and their flow, and capacity constraints on each location and passage of the building are essential. Thus, they must be explicitly defined as given in the following. For each building evacuation problem, safety points serve as the final destinations, which converge into a unique common safety point in set P; the initial locations of evacuees are listed in set R; and the movements of evacuees from one location to another are in set O of candidate activities.

Figure 7 presents an algorithm for transforming a building evacuation problem (BEP) to the corresponding time-expanded process-network synthesis (PNS_T) problem, i.e., algorithm $BEPtoPNS_T$. It generates three classes of evacuees' movements pertaining to [42, 47]. The first class represents the number of evacuees staying at a specific location for at least one unit of time; the second class, the number of evacuees traveling from location i at time t to location j through passage k in time $t + \lambda_{ijk}\Delta t$ where $lim_{\Delta t \to 0} \frac{\lambda_{ijk}\Delta t}{\Delta t} = \lambda_{ijk}$ symbolizes the travel time from i to j[7] through passage k; and, the third class, the number of individuals reaching a common safety point at time t, $0 < t \leq T$.

2.3.3 Algorithm BEPtoPNS$_T$

Figure 8 presents a simple motivational example for illustration to facilitate the comprehension of algorithm $BEPtoPNS_T$ described herein. Algorithm $BEPtoPNS_T$ comprises two mayor parts, the initialization part and the time-expansion part. The initialization part (statements $st1$, $st2$ and loop $lp1$) specifies the set of available raw materials and the set of desired products to be manufactured as well their parameters. The time-expansion part (statement $st3$ and loops $lp2$ and $lp3$) specifies the set of candidates operating units as well their parameters.

For each node $i \in N$ in G (as introduced in *Problem Definition* section , G specifies a conventional building evacuation problem) where ic_i, the initial content of node i, greater than 0 is transformed into raw material r and added to set R (statement $st1$ and loop $lp1$); as such, Axiom (S2) is satisfied, i.e., a vertex of the M-type has no input if and only if it represents a raw material. Algorithm $BEPtoPNS_T$ generates the resources, R_i; R_A, R_B, and R_C. Also, for each resource, R_i, lower bound L_{R_i}, and upper bound U_{R_i}, are set; as such, algorithm $BEPtoPNS_T$ specifies the total amount of available resources for the PNS_T problem. That is, L_{R_i} lower bound of resource R_i, $L_{R_A} = 0$, $L_{R_B} = 0, L_{R_C} = 0$; U_{R_i} upper bound of resource R_i, $U_{R_A} = 3$, $U_{R_B} = 4, U_{R_C} = 3$. Thus, only one product, i.e., $SafetyPoint$, is specified and added to set P (statement $st2$); as such, Axiom (S1) is automatically satisfied, i.e., every final product is represented in the graph. Note that this is

[7] In the literature, classes 1 and 2 are known as holdover and movement arcs, respectively [47].

analogous to the notion of super-sink nodes in maximum-flow problems [25]. In other words, a building-evacuation problem with multiple safety points, i.e., network with multiple sinks, can be converted to the building-evacuation problem with only a single safety point, i.e., network with only a single sink [47]. For outcome $P_{SafetyPoint}$, algorithm $BEPtoPNS_T$ sets lower bound $L_{P_{SafetyPoint}}$, and upper bound $U_{P_{SafetyPoint}}$; as such, the amount of product to be manufactured to meet the demand of the PNS_T problem is specified. To be precise, L_{P_k} lower bound of product P_k, $L_{P_{SafetyPoint}} = U_{R_A} + U_{R_B} + U_{R_C}$, thereby resulting in $L_{P_{SafetyPoint}} = 10$; U_{P_k} upper bound of product P_k, $U_{P_{SafetyPoint}} = \infty$ (refer to Figure 7).

Subsequently, algorithm $BEPtoPNS_T$ stepwisely specifies, in loop $lp2$, the operating units, representing evacuees' movements, as described in the preceding section; as such, Axiom (S3) is satisfied, i.e., every vertex of the O-type represents an operating unit defined in the synthesis problem. First, the algorithm loops through every value of t for $0 \leq t < T$, where t is time. Consequently, for each node $i \in N$ in G, where cap_i, the capacity of node i, is not infinite is transformed into material m and added to set M.

By presuming that an evacuation time $T = 3$, algorithm $BEPtoPNS_T$ generates materials $M_{i\_(t+1)}$; M_{A_1}, M_{B_1}, M_{C_1}, M_{A_2}, M_{B_2}, M_{C_2}, M_{A_3}, M_{B_3}, and M_{C_3}, where $0 \leq t < T$. Note that material $M_{i\_(t+1)}$ represents the number of individuals accommodated at i in time $t + 1$. Then, operating unit o is created and added to set O for each t and i. Algorithm $BEPtoPNS_T$ generates operating units $O_{i\_i\_t\_(t+1)}$; $O_{A\_A\_0\_1}$, $O_{B\_B\_0\_1}$, $O_{C\_C\_0\_1}$, $O_{A\_A\_1\_2}$, $O_{B\_B\_1\_2}$, $O_{C\_C\_1\_2}$, $O_{A\_A\_2\_3}$, $O_{B\_B\_2\_3}$, and $O_{C\_C\_2\_3}$, where $0 \leq t < T$, such that $A\_A\_0\_1 = (\{A\}, \{A\_1\})$, $B\_B\_0\_1 = (\{B\}, \{B\_1\})$, $C\_C\_0\_1 = (\{C\}, \{C\_1\})$, $A\_A\_1\_2 = (\{A\_1\}, \{A\_2\})$, $B\_B\_1\_2 = (\{B\_1\}, \{B\_2\})$, $C\_C\_1\_2 = (\{C\_1\}, \{C\_2\})$, $A\_A\_2\_3 = (\{A\_2\}, \{A\_3\})$, $B\_B\_2\_3 = (\{B\_2\}, \{B\_3\})$, and $C\_C\_2\_3 = (\{C\_2\}, \{C\_3\})$. Additionally, lower bound $L_{O_{i\_i\_t\_(t+1)}}$ and upper bound $U_{O_{i\_i\_t\_(t+1)}}$ are set for each operating unit $O_{i\_i\_t\_(t+1)}$; as such, algorithm $BEPtoPNS_T$ specifies the number of individuals who prefer to stay at specific location i for at least one unit of time, i.e., $t + 1$. Namely, $L_{O_{i\_i\_t\_(t+1)}}$

lower bound of operating unit $O_{i\_i\_t\_(t+1)}$, $L_{O_{A\_A\_0\_1}} = 0$, $L_{O_{B\_B\_0\_1}} = 0$, $L_{O_{C\_C\_0\_1}} = 0$, $L_{O_{A\_A\_1\_2}} = 0$, $L_{O_{B\_B\_1\_2}} = 0$, $L_{O_{C\_C\_1\_2}} = 0$, $L_{O_{A\_A\_2\_3}} = 0$, $L_{O_{B\_B\_2\_3}} = 0$, and $L_{O_{C\_C\_2\_3}} = 0$; $U_{O_{i\_i\_t\_(t+1)}}$ upper bound of operating unit $O_{i\_i\_t\_(t+1)}$, $U_{O_{A\_A\_0\_1}} = 5$, $U_{O_{B\_B\_0\_1}} = 8$, $U_{O_{C\_C\_0\_1}} = 20$, $U_{O_{A\_A\_1\_2}} = 5$, $U_{O_{B\_B\_1\_2}} = 8$, $U_{O_{C\_C\_1\_2}} = 20$, $U_{O_{A\_A\_2\_3}} = 5$, $U_{O_{B\_B\_2\_3}} = 8$, and $U_{O_{C\_C\_2\_3}} = 20$. Subsequently, node j is transformed into material m and added to set M for each arc $(i, j) \in A$ in G, where either cap_j is not infinite or j is the only safety point.

```
input: $G = (N, A), T$
comment: $G = (N, A)$ represents a building evacuation problem, variable $T$ stores the upper bound of the
evacuation time
output: sets $P, R, O$
comment: $R \subset M, P \subset M, R \cap P = \emptyset$, variable $evacuees$ stores the sum of all individuals in the building,
set $building\_exits$ contains all possible gather points in the building, symbol | | represents set cardinality
begin
    comment: initialization part of the algorithm
    st1: $rooms\_with\_people := N \setminus \{x \mid x = \bigcup_{n \in N \wedge ic_n > 0} n\}$
    lp1: while $rooms\_with\_people$ is not empty do
        begin
            let $n$ be an element of $rooms\_with\_people$
            $evacuees +:= ic_n; r := \{n\}; U_r := ic_n; L_r := 0; R := R \cup \{r\}; M := M \cup \{r\};$
            $rooms\_with\_people := rooms\_with\_people \setminus n;$
        end;
    st2: $p := \{SuperExit\};\ U_p := \infty;\ L_p := evacuees;\ P := P \cup \{p\};\ M := M \cup \{p\};$
    comment: time expansion part of the algorithm
    st3: $building\_exits := \{x \mid x = \bigcup_{n \in N \wedge cap_n = \infty} n\};$
    comment: time − expansion part of the algorithm
    lp2: for t := 0 until T − 1 do
        begin
            $building\_locations := N \setminus building\_exits;$
            while $building\_locations$ is not empty do
            begin
                let $i$ be an element of $building\_locations$;
                $M := M \cup \{i\_(t+1)\};$
                if $t = 0$ then begin $i\_i\_t\_(t+1) := \{\{i\}, \{i\_(t+1)\}\}$; end;
                else begin $i\_i\_t\_(t+1) := \{\{i\_t\}, \{i\_(t+1)\}\}$; end;
                $O := O \cup \{i\_i\_t\_(t+1)\};$
                $U_{i\_i\_t\_(t+1)} := cap_i;\ L_{i\_i\_t\_(t+1)} := 0;\ cp_{i\_i\_t\_(t+1)} := 0;$
                $building\_connections := \{x \mid x = \bigcup_{(k,j) \in A \wedge k \neq i} (k, j)\};$
                $room\_connections := A \setminus building\_connections;$
                while $room\_connections$ is not empty do
                begin
                    let $(i, j)$ be an element of $room\_connections$;
                    if $j \cap building_{exits} = \emptyset$ then begin $m := \{j\_(t + \lambda_{ij})\}$; end;
                    else begin
                            if $|building\_exits| = 1$ then begin $m := \{j\_(t + \lambda_{ij})\}$; end;
                            else begin $m = \{Exit\_(t + \lambda_{ij})\}$; end;
                    end;
                    if $t = 0$ then begin $i\_j\_t\_(t + \lambda_{ij}) = \{\{i\}, \{m\}\}$; end;
                    else begin $i\_j\_t\_(t + \lambda_{ij}) = \{\{i\_t\}, \{m\}\}$; end;
                    $M := M \cup \{m\}; O := O \cup \{i\_j\_t\_(t + \lambda_{ij})\};$
                    $U_{i\_j\_t\_(t+\lambda_{ij})} := cap_{(i,j)}; L_{i\_j\_t\_(t+\lambda_{ij})} := 0; cp_{i\_j\_t\_(t+\lambda_{ij})} := 0;$
                    $room\_connections := room\_connections \setminus (i, j);$
                    $building\_locations := building\_locations \setminus i;$
                end;
            end;
        end;
    lp3: for t := 1 until $T$ do
    begin
        if$|building\_exits|! = 1$ then
        begin
            for each $p$ in $building_{exits}$ do
            begin $evactime\_t := \{\{p_t\}, \{SuperExit\}\}$; end;
        end;
        else begin $evactime_t := \{\{Exit_t\}, \{SuperExit\}\}$; end;
        $U_{evactime\ \_t} := \infty;\ L_{evactime\ \_t} := 0;\ cp_{evactime\ \_t} := t;\ O := O \cup \{evactime\_t\};$
    end;
end;
```

Figure 7. Algorithm $BEPtoPNS_T$ written in Pidgin Algol (see Appendix C).

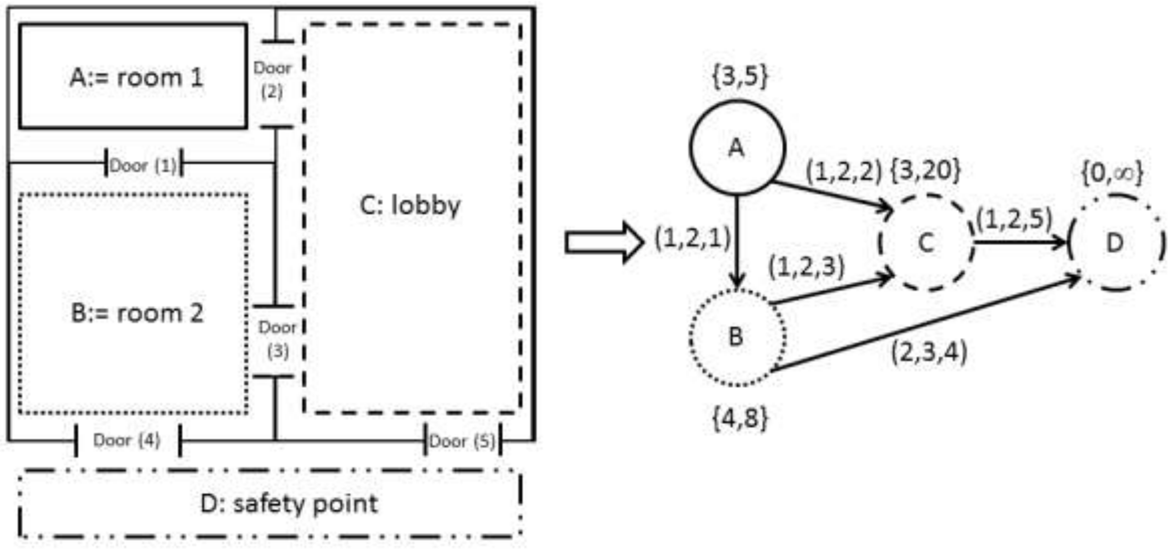


Figure 8. Motivational example for illustration: {ic_i initial content of node i, cap_i capacity of node i}; (λ_{ij} travel time from node i to node j, $cap(i,j,k)$ capacity of arc from node i to j through section k, which connects locations i and j).

Algorithm $BEPtoPNS_T$ generates materials $M_{j\_(t+\lambda_{ijk})}$[8];M_{D_1}, M_{D_2}, M_{D_3}, and M_{D_4}; where $0 \le t < T$. Next, for each t, i, and j, operating unit o is created and added to set O. Algorithm $BEPtoPNS_T$ generates operating units $O_{i\_j\_t\_(t+\lambda_{ijk})\_k}$; $O_{A\_B\_0\_1\_1}$, $O_{A\_C\_0\_1\_2}$, $O_{B\_C\_0\_1\_3}$, $O_{B\_D\_0\_2\_4}$, $O_{C\_D\_0\_1\_5}$, $O_{A\_B\_1\_2\_1}$, $O_{A\_C\_1\_2\_2}$, $O_{B\_C\_1\_2\_3}$, $O_{B\_D\_1\_3\_4}$, $O_{C\_D\_1\_2\_5}$, $O_{A\_B\_2\_3\_1}$, $O_{A\_C\_2\_3\_2}$, $O_{B\_C\_2\_3\_3}$, $O_{B\_D\_2\_4\_4}$, and $O_{C\_D\_2\_3\_5}$; where $0 \le t < T$, such that $O_{A\_B\_0\_1\_1} = (\{A\},\{B\_1\})$, $O_{A\_C\_0\_1\_2} = (\{A\},\{C\_1\})$, $O_{B\_C\_0\_1\_3} = (\{B\},\{C\_1\})$, $O_{B\_D\_0\_2\_4} = (\{B\},\{D\_2\})$, $O_{C\_D\_0\_1\_5} = (\{C\},\{D\_1\})$, $O_{A\_B\_1\_2\_1} = (\{A\_1\},\{B\_2\})$, $O_{A\_C\_1\_2\_2} = (\{A\_1\},\{C\_2\})$, $O_{B\_C\_1\_2\_3} = (\{B\_1\},\{C\_2\})$, $O_{B\_D\_1\_3\_4} = (\{B\_1\},\{D\_3\})$, $O_{C\_D\_0\_1\_5} = (\{C\},\{D\_1\})$, $O_{A\_B\_2\_3\_1} = (\{A\_2\},\{B\_3\})$, $O_{A\_C\_2\_3\_2} = (\{A\_2\},\{C\_3\})$, $O_{B\_C\_2\_3\_3} = (\{B\_2\},\{C\_3\})$, $O_{B\_D\_2\_4\_4} = (\{B\_2\},\{D\_4\})$, and $O_{C\_D\_2\_3\_5} = (\{C\_2\},\{D\_3\})$. addition, lower bound $L_{O_{i\_j\_t\_(t+\lambda_{ijk})\_k}}$ and upper bound $U_{O_{i\_j\_t\_(t+\lambda_{ijk})\_k}}$ are set for each

[8] If j represents one of many safety points of a building, then, algorithm $BEPtoPNS_T$ generates the materials $M_{Exit\_(t+\lambda_{ijk})}$.

operating unit $O_{i\_j\_t\_(t+\lambda_{ijk})\_k}$; as such, algorithm $BEPtoPNS_T$ specifies the number of evacuees traveling from location i at time t to location j through passage k in time $t+\lambda_{ijk}$. That is, $L_{O_{i\_j\_t\_(t+\lambda_{ij})\_k}}$ lower bound of operating unit $O_{i\_j\_t\_(t+\lambda_{ij})\_k}$, $L_{O_{A\_B\_0\_1\_1}} = 0$, $L_{O_{A\_C\_0\_1\_2}} = 0$, $L_{O_{B\_C\_0\_1\_3}} = 0$, $L_{O_{C\_D\_0\_1\_5}} = 0$, $L_{O_{B\_D\_0\_2\_5}} = 0$, $L_{O_{A\_B\_1\_2\_1}} = 0$, $L_{O_{A\_C\_1\_2\_2}} = 0$, $L_{O_{B\_C\_1\_2\_3}} = 0$, $L_{O_{C\_D\_1\_2\_5}} = 0$, $L_{O_{B\_D\_1\_3\_4}} = 0$, $L_{O_{A\_B\_2\_3\_1}} = 0$, $L_{O_{A\_C\_2\_3\_2}} = 0$, $L_{O_{B\_C\_2\_3\_3}} = 0$, $L_{O_{C\_D\_2\_3\_5}} = 0$, and $L_{O_{B\_D\_2\_4\_4}} = 0$; $U_{O_{i\_j\_t\_(t+\lambda_{ij})\_k}}$ upper bound of operating unit $O_{i\_j\_t\_(t+\lambda_{ij})\_k}$, $U_{O_{A\_B\_0\_1\_1}} = 2$, $U_{O_{A\_C\_0\_1\_2}} = 2$, $U_{O_{B\_C\_0\_1\_3}} = 2$, $U_{O_{B\_D\_0\_2\_4}} = 3$, $U_{O_{C\_D\_0\_1\_5}} = 2$, $U_{O_{A\_B\_1\_2\_1}} = 2$, $U_{O_{A\_C\_1\_2\_2}} = 2$, $U_{O_{B\_C\_1\_2\_3}} = 2$, $U_{O_{B\_D\_1\_3\_4}} = 3$, $U_{O_{C\_D\_1\_2\_5}} = 2$, $U_{O_{A\_B\_2\_3\_1}} = 2$, $U_{O_{A\_C\_2\_3\_2}} = 2$, $U_{O_{B\_C\_2\_3\_3}} = 2$, $U_{O_{B\_D\_2\_4\_4}} = 3$, and $U_{O_{C\_D\_2\_3\_5}} = 2$ (refer to Figure 8).

Finally, algorithm $BEPtoPNS_T$ specifies the operating units, which represent the number of evacuees reaching a common safety point, in loop $lp3$, i.e. $SafetyPoint$; as such, Axioms (S3) and (S4) are satisfied, i.e., every vertex of the O-type represents an operating unit defined in the synthesis problem and every vertex of the O-type has at least one path leading to a vertex of the M-type representing a final product, respectively. First, algorithm $BEPtoPNS_T$ loops through every value of t for $0 < t \leq T$. Hence, one operating unit o is created and added to set O for each t. Algorithm $BEPtoPNS_T$ generates the operating units, $O_{evactime\_t}$; $O_{evactime\_1}$, $O_{evactime\_2}$, and $O_{evactime\_3}$; where $0 < t \leq T$, such that $O_{evactime\_1} = (\{D\_1\}, \{SafetyPoint\})$, $O_{evactime\_2} = (\{D\_2\}, \{SafetyPoint\})$, and $O_{evactime\_3} = (\{D\_3\}, \{SafetyPoint\})$[9]. Moreover, lower bound $L_{O_{evactime\_t}}$, upper bound $U_{O_{evactime\_t}}$, and proportional cost $cp_{O_{evactime\_t}}$ for each operating unit $O_{evactime\_t}$ are set; as such, algorithm $BEPtoPNS_T$ specifies the number of evacuees reaching a safety point in time t. Specifically, $L_{O_{evactime\_t}}$ lower bound of operating unit $O_{evactime\_t}$, $L_{O_{evactime\_1}} = 0, L_{O_{evactime\_2}} = 0$, and $L_{O_{evactime\_3}} = 0$; $U_{O_{evactime\_t}}$ upper bound of operating unit $O_{evactime\_t}$, $U_{O_{evactime\_1}} = \infty, U_{O_{evactime\_2}} = \infty$, and $U_{O_{evactime\_3}} = \infty$; $cp_{O_{evactime\_t}}$ proportional cost of operating unit $O_{evactime\_t}$, $cp_{O_{evactime\_1}} = 1, cp_{O_{evactime\_2}} = 2$, and

[9] If the building has more than one safety point, algorithm $BEPtoPNS_T$ generates, the operating units $O_{evactime\_t}$ for $0 < t \leq T$, such that $O_{evactime\_t} = (\{Exit\_t\}, \{SafetyPoint\})$.

$cp_{O_{evactime\_3}} = 3$ (refer to Figure 10). As result, the execution of loops $lp2$ and $lp3$ assures that Axiom (S5) is satisfied by the maximal structure, i.e., 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 9 displays the maximal structure of the motivational example generated by algorithm MSG. Furthermore, Figure 10 shows the relationships between the elements adopted in the definition of a conventional building evacuation problem and those adopted in the specification of a time-expanded process-network synthesis problem.

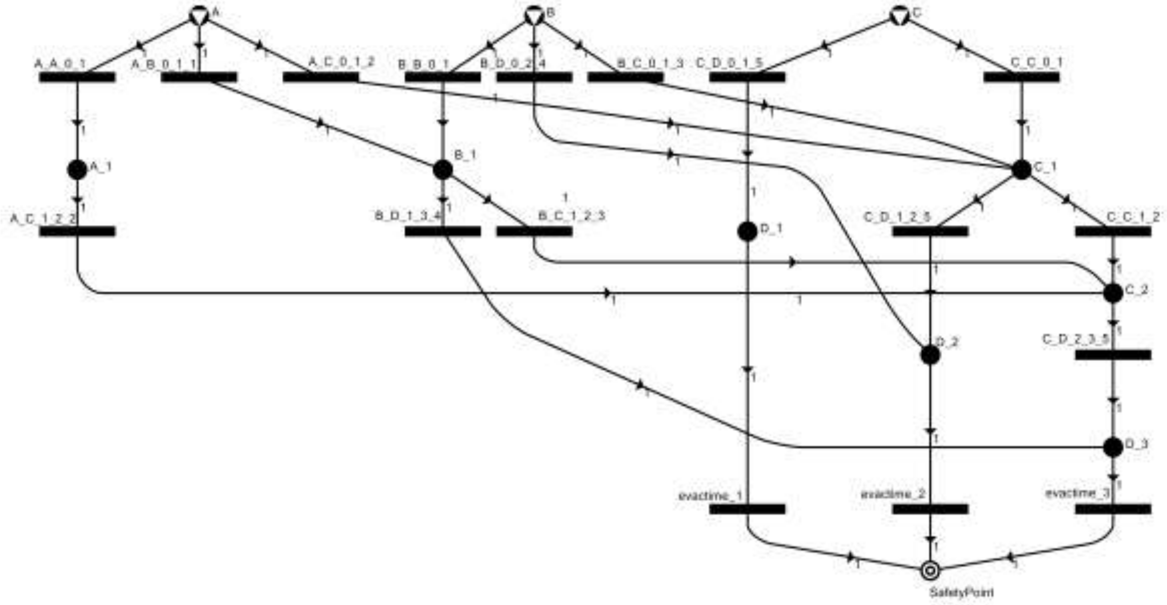


Figure 9. Maximal structure of the motivational example.

2.3.4 Mathematical programming model

The mathematical programming model derived from the maximal structure, generated by algorithm MSG, should be as simple as possible without impairing the optimality of the resultant solution. In any of the available algorithmic methods for addressing evacuation problems, the model derived leads to a linear mathematical programming problem [47,100]. This linear programming model is formulated in terms of a dynamic network flow, and then solved by applying any minimum cost static network flow algorithm [1,47].

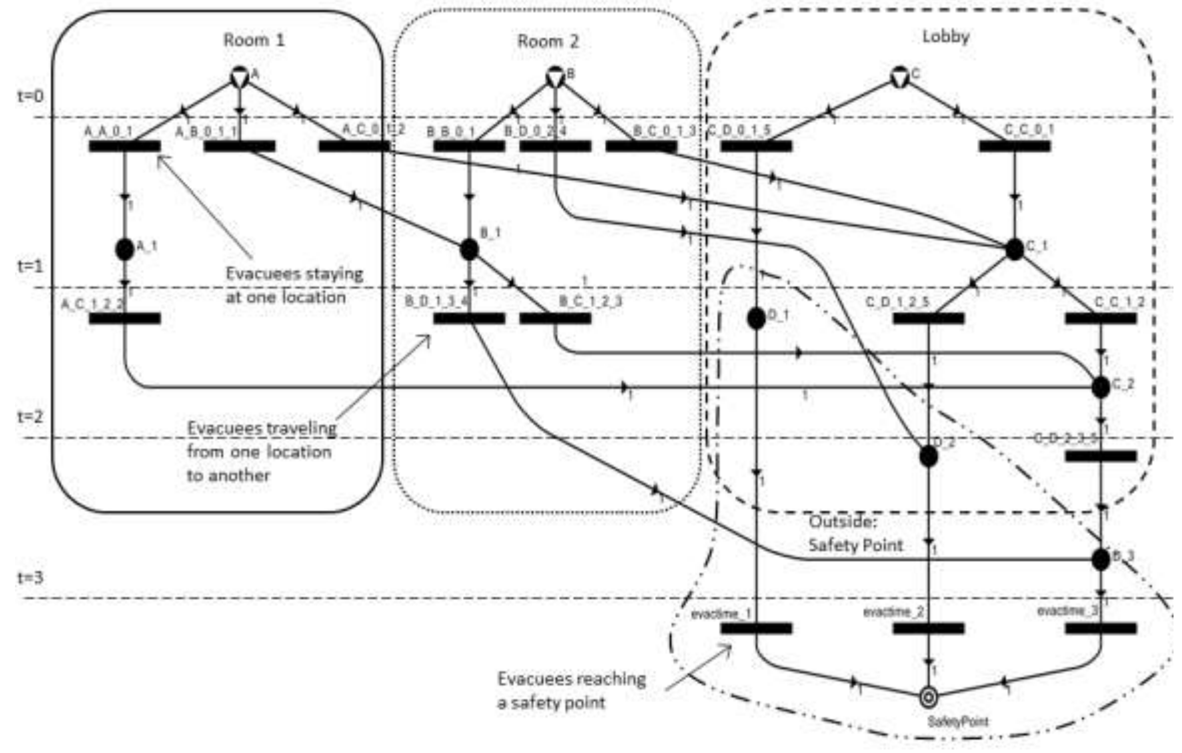

Room 1
Room 2
Lobby
t=0
t=1
t=2
t=3
Evacuees staying at one location
Evacuees traveling from one location to another
Outside: Safety Point
Evacuees reaching a safety point
SafetyPoint
evactime_1
evactime_2
evactime_3

Figure 10. Maximal structure of the motivational example showing the relationships between the elements adopted in the definition of a BEP and those adopted in the specification of a PNS_T.

In the present work, a mixed-integer linear programming ($MILP$) model has been formulated below, which at the very least yields a solution identical with those conventional network flow algorithms [47,70,72].

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 $O = \wp(M) x \wp(M)$[10]. 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 proportional cost cp_{O_i} (refer to $lp3$ in Figure 7).

[10] $\wp$ represents powerset.

Moreover, lower bound L_{R_i} and upper bound U_{R_i} 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 $y_{O_i} = 1$, 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, $x_{O_i} \leq y_{O_i} U_i$, 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, minimal, is the objective value. The resultant $MILP$ model is given in the following.

$$z = min\left(\sum_{O_i \in O} cp_{O_i} * x_{O_i}\right) \tag{2.1}$$

subject to (2.2)

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

$$0 \leq x_{O_i}, L_{O_i} \leq x_{O_i} \leq U_{O_i} \qquad \forall O_i \in O \tag{2.3}$$

$$L_{P_j} \leq \sum_{O_i \in O} a_{ij} * x_{O_i} \leq U_{P_j} \qquad \forall P_j \in M \cap P \tag{2.4}$$

$$L_{R_j} \leq \sum_{O_i \in O} a_{ji} * x_{O_i} \leq U_{R_j} \qquad \forall R_j \in M \cap R \tag{2.5}$$

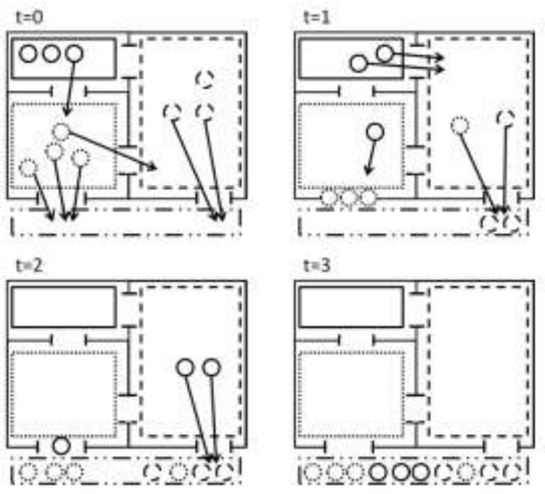
$$L_{M_j} \leq \sum_{O_i \in O} a_{ji} * x_{O_i} - \sum_{O_i \in O} a_{ij} * x_{O_i} \leq U_{M_j} \qquad \forall M_j \in M \backslash (R \cup P) \tag{2.6}$$

$$x_{O_i} \leq y_{O_i} L \tag{2.7}$$

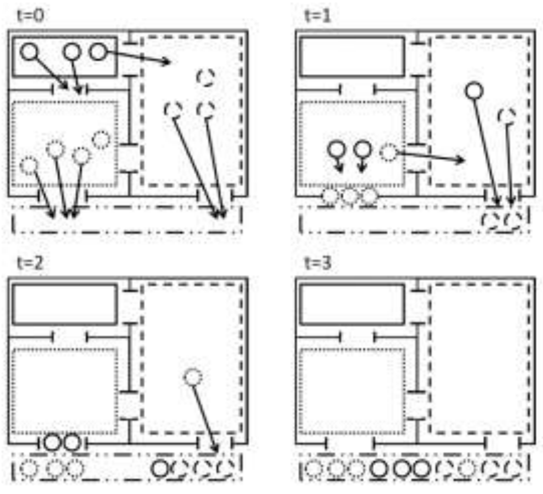
$$y_{O_i} \in \{0,1\} \tag{2.8}$$

The maximal structure serves as the input to the generation and solution of the MILP model by algorithm ABB [35]. It yields the optimal network and a finite number of *n*-best suboptimal networks in ranked order, whenever computationally possible, due to the complexity of the evacuation problem [100]. Algorithm ABB has found a total of 10 feasible evacuation routes in less than a second, i.e., 0.080 s, on an Intel(R) Core(R) i5 CPU 650 @ 3.20 GHz.

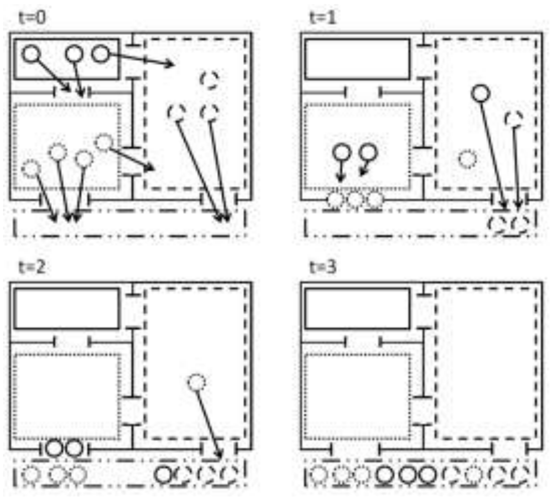
Figure 11 shows four of them for the example of $T = 3$. Algorithms MSG and ABB have been executed by software PNS Studio [93].



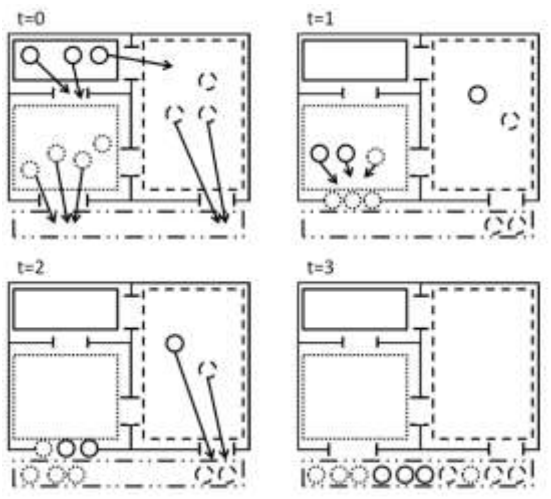
(a)



(b)



(c)



(d)

Figure 11. Solution #1 (a), #2 (b), #3 (c), and #4 (d) obtained via Algorithm ABB.

2.4 Results and Discussion

The alternative feasible evacuation routes generated by algorithm ABB are ranked according to several criteria, e.g., egress time, average number of periods for an evacuee to evacuate, average number of evacuees per time period, node clearing time, individuals waiting at the end of a time period by node, etc. [49]. For instance, the total evacuation time required to completely evacuate all the individuals is 3 units of time by adopting any of the four evacuation routes generated by algorithm ABB (see Section 3.3). Nonetheless, if evacuation route #1 is compared with evacuation route #4, it can be noted that seven individuals would reach safety point *D* within 2 units of time by employing evacuation route #1, while only five individuals would reach safety point *D* within the same time by employing evacuation route #4. Even if the evacuees leave their initial locations after the onset of a fire emergency may not be considered a decisive factor for any optimal evacuation plan, because other factor such as personal behaviors, e.g., waiting action, could be determinant in an optimal evacuation plan; see Solutions #1 and #4 in Table 1. These observations show that a trade-off analysis and in-depth assessment among different evacuation routes is required [14,50]. Additionally, there are other criteria based upon the effects of fire into the evacuation route planning that should be adopted in the proposed method, e.g., individual travel and exposure time; time-based risk and evacuation exposure; time-space-based risk and evacuation exposure [14,49,50].

Additionally to the one-story building example, other building configurations are studied: two-story [94], three-story [59], and eleven-story [13]. When our method is applied to the two-story building see Figure 12, algorithm ABB has found a total of 347 feasible evacuation routes in nearly 18 min. It is noteworthy to mention that algorithm ABB has found the first 71-best sub-optimal solutions in less than 18 s. This outcome outperforms current optimization models (i.e., EVACNET4, WAYOUT, PathFinder), because not only the optimal solution is computed but the *n*-best sub-optimal solutions are. Table 2 and Figure 13 summarize the results of a subset of four *n*-best sub-optimal solutions. Also, regarding the three-story and eleven-story building, the algorithm ABB has found the 10-best sub-optimal solutions in 1 m 15 s and 17 m 30 s for the three-story building and the eleven-story building respectively (see Table 4 and Table 4; and Figures Figure 14 and Figure 15). These results clearly show the potential of the ABB algorithm for computing the n-best sub-optimal evacuation routes.

Nevertheless, it is important to emphasize that even though the evacuation model size increases proportionally to both the number of discrete locations and the discrete time slots, the proposed method, i.e., formulating the evacuation problem as process network synthesis problem with appropriate targets, and software could generate the optimal solution by solving a single Linear Programming (LP) problem in polynomial time (LP is proven to be solvable in polynomial time). Moreover, ABB is the only available tool capable of generating the *n*-best alternative solutions systematically. Also, the process of generating feasible evacuations plans is always convoluted: it is combinatorial in nature [100]. Finally, it is worth emphasizing that our method: (i) can generate with dispatch the evacuation model and its optimal solution (in polynomial time); and (ii) can generate alternative evacuation plans algorithmically; no similar approaches are capable of doing so [70,72]. Nevertheless, depending on the size of the problem, it might require substantially larger computing time to generate the *n*-best sub-optimal solutions. This may entail the deployment of modern computing techniques, e.g., Grid Computing and High Performance Computing (HPC) [49], to greatly accelerate the computation.

Table 1. Four best assignments for the one-story building example.

Solution	Egress Time	Average # of time period for an evacuee to evacuate	Node Clearing Time			People waiting at the end of a time period by node			
			Nodes			Time	Nodes		
			A	B	C		A	B	C
#1	3	2,1	2	3	3	0	2	0	1
						1	0	0	0
						2	0	0	0
						3	0	0	0
#2	3	2,1	1	2	3	0	0	1	1
						1	0	0	0
						2	0	0	0
						3	0	0	0
#3	3	2,1	1	2	3	0	0	0	1
						1	0	0	1
						2	0	0	0
						3	0	0	0
#4	3	2,3	1	2	3	0	0	1	1
						1	0	0	2
						2	0	0	0
						3	0	0	0

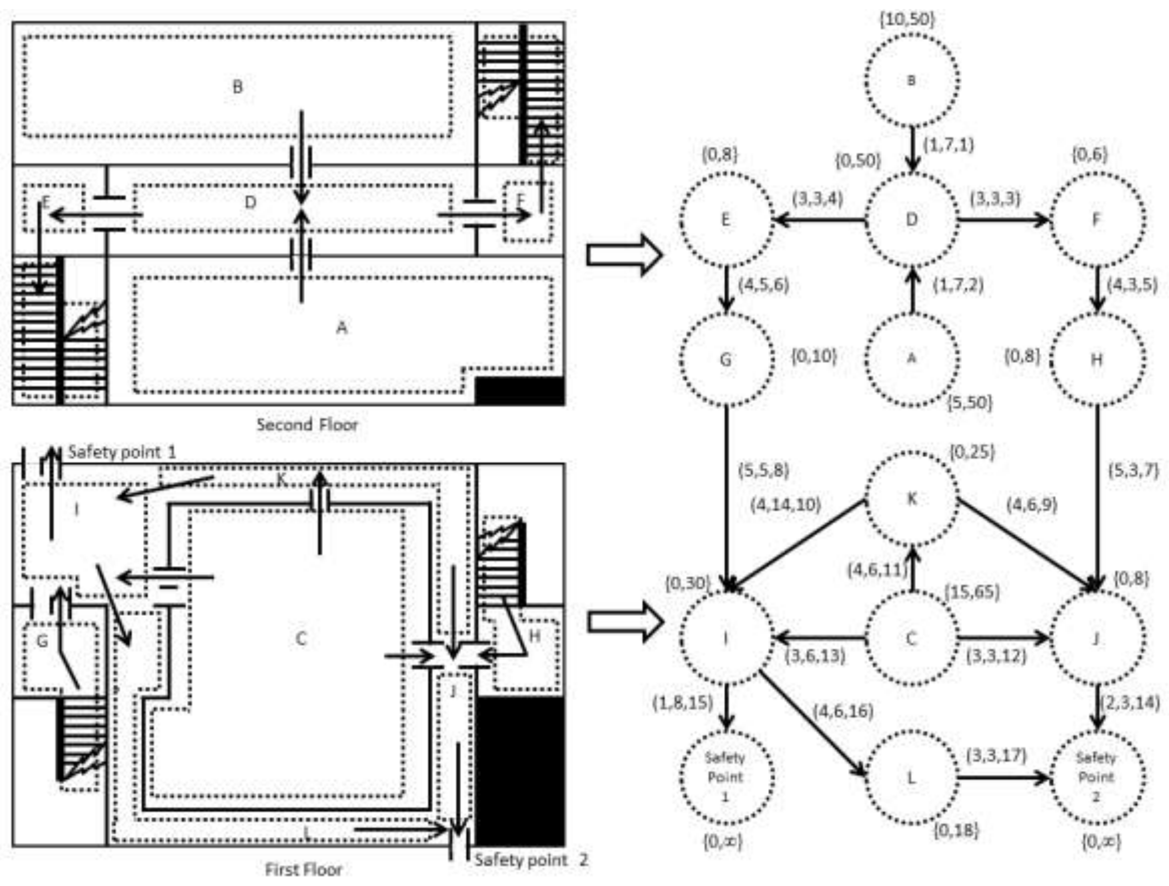


Figure 12. A two-story building floor-map (adapted from [22] and [23]).

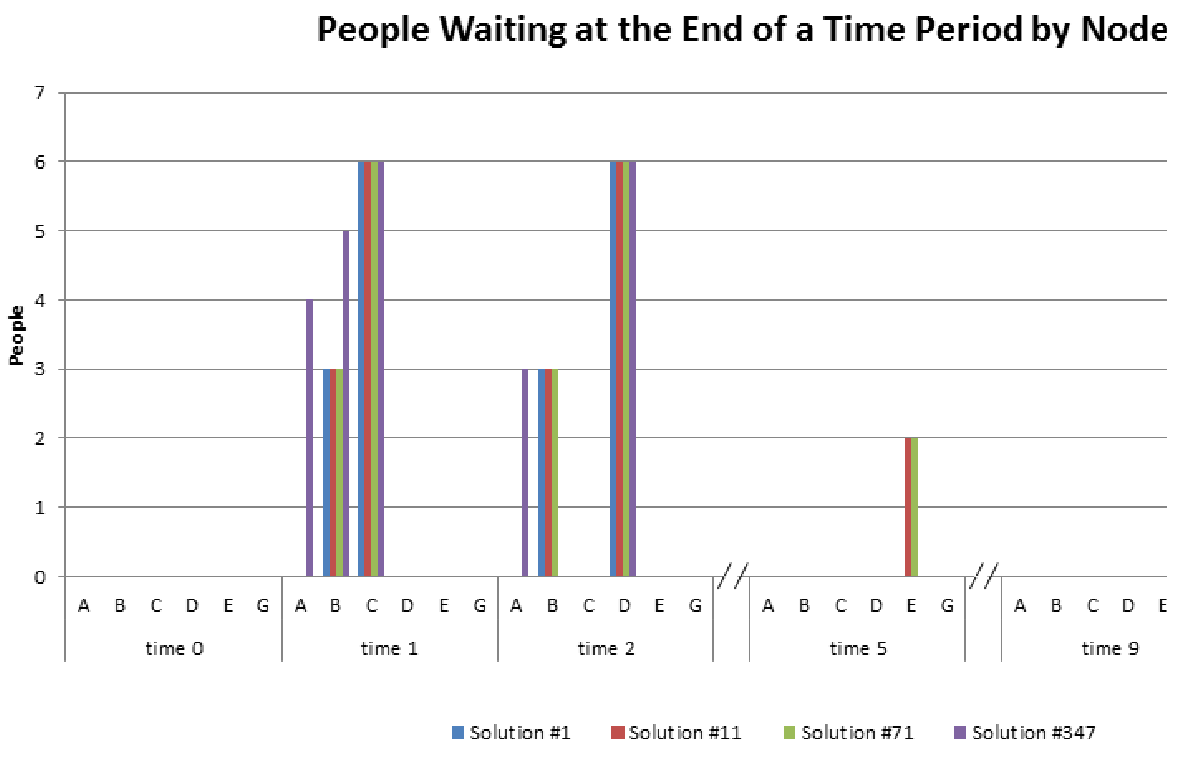


Figure 13. People waiting at the end of a time period by node for the two-story building example. Contents are zero for non-listed nodes and time periods.

Table 2. Four sub-optimal evacuation plans for the two-story building example.

Solution	Egress Time	Average # of time period for an evacuee to evacuate	Node Clearing Time (*)											
			Nodes											
			A	B	C	D	E	F	G	H	I	J	K	L
#1	16	9,9	**0**	**2**	**1**	3	6	5	10	9	15	14	-	-
#11	16	9,97	**0**	**2**	**1**	3	6	5	10	9	15	14	-	-
#71	16	10,03	**0**	**2**	**1**	3	6	5	10	9	15	14	-	-
#347	16	10,1	**2**	**1**	**2**	3	6	5	10	9	15	14	-	-

(*) Nodes K and L are never traversed by employing any of the evacuation plans

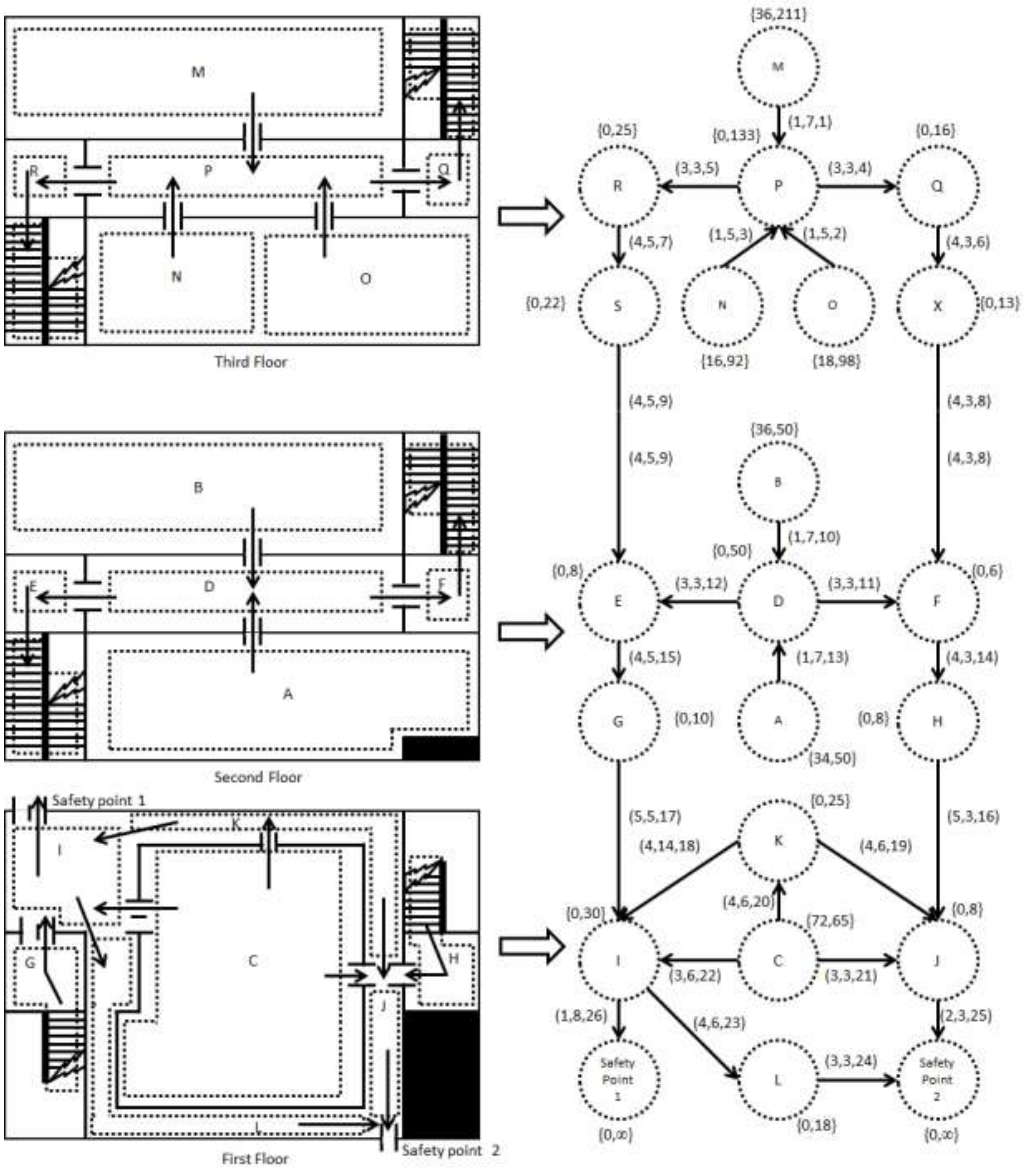


Figure 14. A three-story building floor-map (adapted from [22] and [23]).

Table 3. Data taken from the best optimal solution for the three-story building example.

	PNS Solver	PNS Configuration File		P-graph Model Size		Elements excl
T	Computation Time (s)	Input Size (KB)	Output Size (KB)	# Materials	# Operating Units	% Materials
…	…	…	…	…	…	…
34	0,062	140	108	427	1138	45,12
35	0,068	144	110	448	1194	43,93
36	0,086	148	112	469	1250	42,8
37	0,075	153	115	490	1306	41,74
38	0,074	157	117	511	1362	40,72
39	0,074	161	119	532	1418	39,75
40	0,077	165	122	553	1474	38,83
41	0,079	170	124	574	1530	37,95
42	0,087	174	126	595	1586	37,1
43	0,09	178	128	616	1642	36,3
44	0,095	183	131	637	1698	35,53
45	0,092	187	133	658	1754	34,79
46	0,092	191	136	679	1810	34,08
47	0,1	196	138	700	1866	33,4
48	0,096	200	140	721	1922	32,74
49	0,097	204	142	742	1978	32,11
50	0,096	208	144	763	2034	31,51

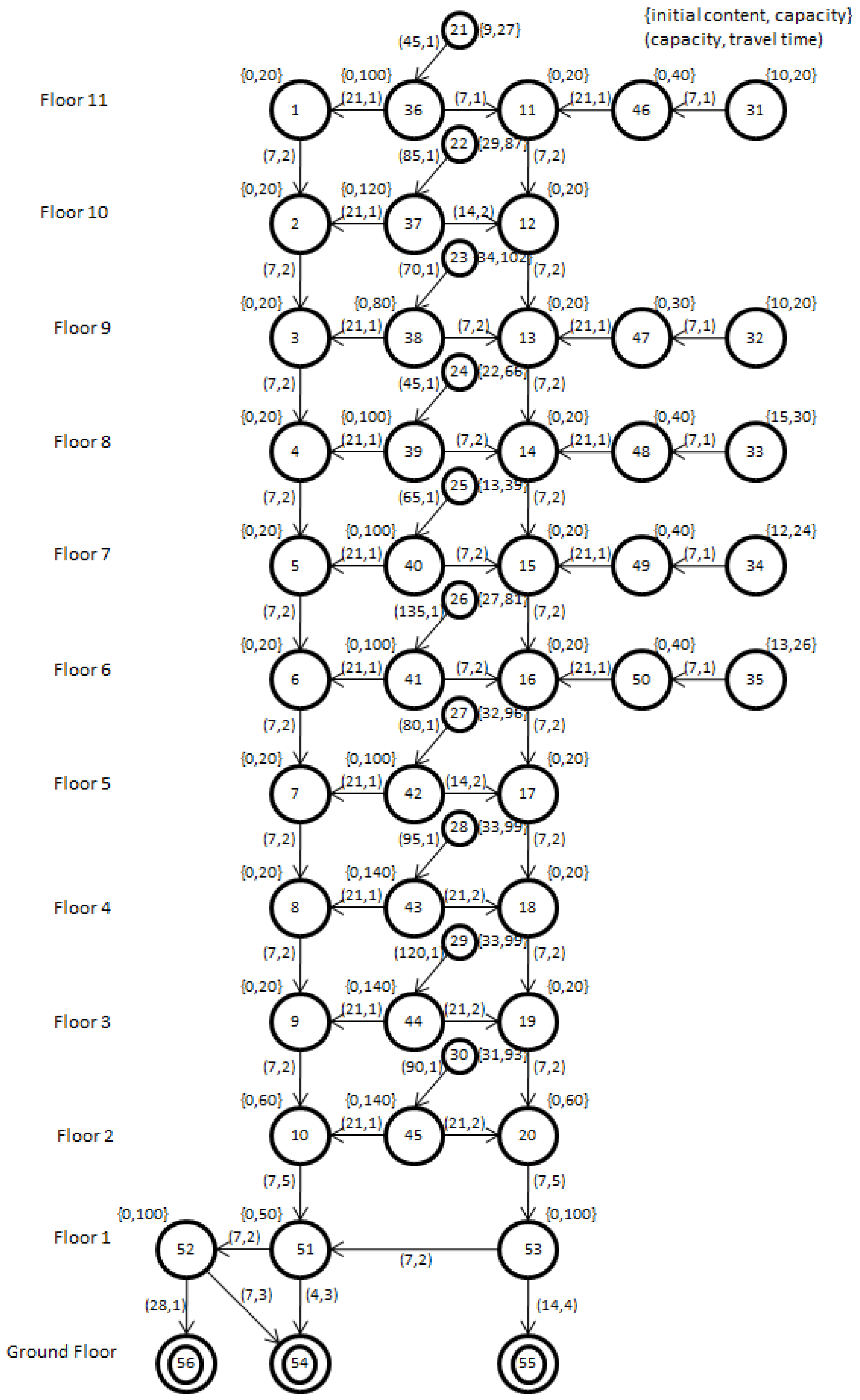


Figure 15. An eleven-story building floor-map (adapted from [10]).

Table 4. Data taken from the best optimal solution for the eleven-story building example.

	PNS Solver	PNS Configuration File		Pgraph Model Size		MSG Al
T	Computation Time (s)	Input Size (KB)	Output Size (KB)	# Materials	# Operating Units	% Materi
…	…	…	…	…	…	…
34	0,151	359	264	819	1768	57,37
35	0,164	370	271	873	1888	55,8
36	0,18	381	278	927	2008	54,31
37	0,175	392	282	981	2128	52,9
38	0,194	403	291	1035	2248	51,57
39	0,188	413	296	1089	2368	50,3
40	0,235	424	305	1143	2488	49,09
41	0,192	435	307	1197	2608	47,93
42	0,202	446	316	1251	2728	46,83
43	0,225	457	320	1305	2848	45,78
44	0,228	468	330	1359	2968	44,78
45	0,224	479	335	1413	3088	43,82
46	0,237	490	342	1467	3208	42,9
47	0,243	501	350	1521	3328	42,01
48	0,233	511	353	1575	3448	41,17
49	0,27	522	360	1629	3568	40,35
50	0,25	533	366	1683	3688	39,57

Chapter 3. Designing Organization-based Multiagent Systems: Research Results

3.1 Background

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 [18]. Trends in the field of autonomous agents and multiagent systems suggest that the explicit design and use of organization-based multiagent systems [76], which allow heterogeneous agents (either human or artificial entities) rely on well-defined roles to accomplish either individual or system level goals [21,114], is a promising approach to these new requirements [76]. In the literature a set of methodologies [52], a selection of design processes [15], and a collection of frameworks [18,20,24,26,55,65,99] are available to provide the basis for constructing sophisticated autonomous multi-agent 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 multi-agent organizations [56,63,88,95].

Nevertheless, in situations where the nature of the environment makes the organization susceptible to individual failures, these failures could significantly reduce the ability of the organization to accomplish its goals. The above-mentioned methodologies and frameworks, however, do not offer techniques for identifying the number of feasible configurations of agents 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) [81]. 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 [65,81].

Following, two algorithmic methods for assessing the design of organization-based multiagent systems supported by software tools at each step are discussed. These method resorts to the graph-theoretic approach based on the P-graph framework. The methods are demonstrated by applying them to different organization-based multiagent system designs.

3.2 Problem Definition

Given a design of an organization-based multiagent system, O_{OMACS}, comprising a set of heterogeneous agents, we are to investigate the number of feasible configurations of agents that can be synthesized, or can emerge. Moreover, we are to explore the reliability and cost of the system with such configurations.

In the current dissertation, a set of heterogeneous agents is defined as a set of cooperative entities – either human or artificial (hardware or software) – capable of perceiving and acting on their environment with the purpose of achieving their design goals. We measure a system's feasibility in light of its possibility of implementation; the system's flexibility in view of its ability to overcome the individual elements' (agents') failures; and finally, the system's cost in terms of the total cost of the agents therein.

3.3 Methodology

3.3.1 Designing Organization-based Multiagent Systems by resorting to the framework OMACS

To demonstrate the application of the P-graph framework for assessing the designs of organization-based multi-agent system, a survey is given of a simplified Cooperative Robotic Search Team (CRST) system [51,95]. 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 [38].

3.3.1.1 Overview of the CRST Organization

For illustration, it is presumed that four goals be achieved by the CRST. In other words, $G_{OMACS} = \{g_1, g_2, g_3, g_4\}$ where g_i for $1 \leq i \leq 4$ signifies "search area i." In the CRST, two roles are identified, i.e., $R_{OMACS} = \{r_1, r_2\}$ 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 $achieves = \{(r_1, g_1, 0.2),$ $(r_1, g_2, 0.4), (r_1, g_3, 0.6), (r_1, g_4, 0.8), (r_2, g_1, 1.0), (r_2, g_2, 0.7), (r_2, g_3, 0.4), (r_2, g_4, 0.1)\}$.

Also, four capabilities are specified, i.e., $C_{OMACS} = \{c_1, c_2, c_3, c_4\}$. 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, i.e., $A_{OMACS} = \{a_1, a_2, a_3\}$; 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: $possesses = \{(a_1, c_1, 0.3), (a_1, c_2, 0.5), (a_1, c_3, 0.5),$ $(a_1, c_4, 0.3), (a_2, c_2, 0.7), (a_2, c_3, 0.5), (a_2, c_4, 0.7), (a_3, c_2, 0.4), (a_3, c_3, 0.9), (a_3, c_4, 0.2)\}$.

Additionally, the cost of each individual agent a_1, a_2, and a_3 is \$850, \$900, and \$950; respectively (see Figure 16).

3.3.2 Algorithm OMACStoPNS

Algorithm $OMACStoPNS$ comprises two mayor parts, the initialization and the construction parts. The initialization part (statements $st1$, $st2$, $st3$, and loops $lp1$ and $lp2$) 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 17).

Each agent a_i in A is transformed into raw material r to be added to set R (loop $lp1$); as such, Axiom (S2) is satisfied, i.e., a vertex of the M-type has no input if and only if it represents a raw material. 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 5). 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, i.e., every final product is represented in the graph. 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 $\langle a_i, r_k, g_j \rangle$ indicates that agent $a_i \in A$ has been assigned to play role $r_k \in R$ in order to achieve goal $g_j \in G$. 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 6).

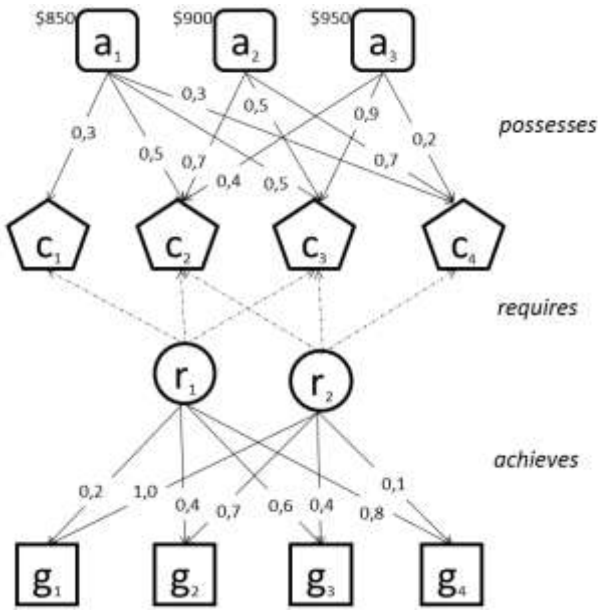


Figure 16. Overview of the CRST Organization. The boxes at the top of the diagram represent agents identified by their types, the circles represent the roles, the pentagon's represent capabilities, and the squares are system's goals. The arrows between the entities represent *achieves*, *requires*, and *possesses* functions/relations.

Subsequently, algorithm $OMAStoPNS$ stepwisely specifies the operating units in loops $lp2$ and $lp3$, representing organizational assignments (see Figure 17); as such, Axioms (S3) and (S4) are satisfied, , i.e., every vertex of the O-type represents an operating unit defined in the synthesis problem and every vertex of the O-type has at least one path leading to a vertex of the M-type representing a final product, respectively. First, the algorithm loops through every goal $g_j \in G$. Each goal g_j is transformed into material m for inclusion in set M. Algorithm $OMAStoPNS$ 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 $OMAStoPNS$ generates operating units $O_{g_{j\_oaf}}$. 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 7).

```
input: $G, A, R_{OMACS}, achieves, requires, possesses$
comment: $G$ defines the goals of the organizations, $R_{OMACS}$ defines a set of roles, $A$ is a set of agents
$achieves$ defines the extent of achievement of a goal by a role, $(G \times R_{OMACS} \rightarrow [0 \dots 1])$
$possesses$ defines the quality of an agent´s capability $(A \times C \rightarrow [0 \dots 1])$, and $requires$ defines
the set of capabilities required to play a role $(R_{OMACS} \rightarrow \wp(C))$. The $capable$ function
$(A \times R_{OMACS} \rightarrow [0 \dots 1])$ is computed as defined in Eq. 1 in Section 3.
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 $a_i \in A$ do
      begin
          $R := R \cup \{a_i\}; M := M \cup \{a_i\}; U_{a_i} := \infty; L_{a_i} := 0; p_{a_i} := 0$;
      end;
comment: construction part of the algorithm;
lp2:  for $g_i \in G$ do
      begin
          $M := M \cup \{g_i\}; g_{i\_oaf} := \{\{g_i\}, \{oaf\}\}; O := O \cup \{g_{i\_}oaf\}$;
          $U_{g_{i\_oaf}} := \infty; L_{g_{i\_oaf}} := 0; c_{g_{i\_oaf}} := 0$;
          $a_{g_i, g_{i\_oaf}} := 1; a_{g_{i\_oaf}, oaf} := 1$;
      end;
lp3:  for $a_i \in A$ 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 $\wp(c) \subseteq Capabilities_{a_i}$ then
                  $aux := \emptyset$;
                  for $\langle r'', g_j, value'' \rangle \in Achieves$ do
                  begin
                      if $r_k = r''$ then
                          $M := M \cup \{a_{i\_}r_{k\_}g_j\};\ a_{i\_}r_{k\_}g_j := \{\{a_{i\_}r_{k\_}g_j\}, \{g_j\}\}$;
                          $aux := aux \cup \{a_{i\_}r_{k\_}g_j\};\ O := O \cup \{a_{i\_}r_{k\_}g_j\}$;
                          $U_{a_{i\_}r_{k\_}g_j} := \infty; L_{a_{i\_}r_{k\_}g_j} := 0; c_{a_{i\_}r_{k\_}g_j} := 0$;
                          $a_{a_{i\_}r_{k\_}g_j, a_{i\_}r_{k\_}g_j} := 1; a_{a_{i\_}r_{k\_}g_j, g_j} := value''$;
                          $a_{a_i, a_{1\_}r_k} = 1; a_{a_{i\_}r_k, a_{i\_}r_{k\_}g_j} := capable(a_i, r_k)$;
                  end;
                  if $aux\ is\ not\ empty$ then
                      $a_{i\_}r_k := \{\{a_i\}, \{aux\}\}; O := O \cup \{a_{i\_}r_k\}$;
                      $U_{a_{i\_}r_k} := 1; L_{a_{i\_}r_k} := 0; c_{a_{i\_}r_k} := 0$;
                  end;
          end;
      end;
end;
```

Figure 17. Algorithm *OMACStoPNS* written in Pidgin Algol (see Appendix C).

Table 5. Resources to be considered in process synthesis for the example

Resource R_{j}	Lower bound $L_{R_{\mathrm{j}}}$	Upper bound $U_{R_{\mathrm{j}}}$	Cost $c_{R_{\mathrm{j}}}$
a_1	0	∞	850
a_2	0	∞	900
a_3	0	∞	950

Table 6. Targets to be considered in process synthesis for the example

Target P_{j}	Lower bound $L_{P_{\mathrm{j}}}$	Upper bound $U_{P_{\mathrm{j}}}$	Cost $c_{P_{\mathrm{j}}}$
oaf	0	∞	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 $O_{a_i\_r_k\_g_j}$; 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 7). As a result, the execution of loop $lp3$ assures that Axiom (S5) is satisfied by the maximal structure, , i.e., 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 18 displays the maximal structure of the motivational example generated by algorithm MSG.

Table 7. 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.

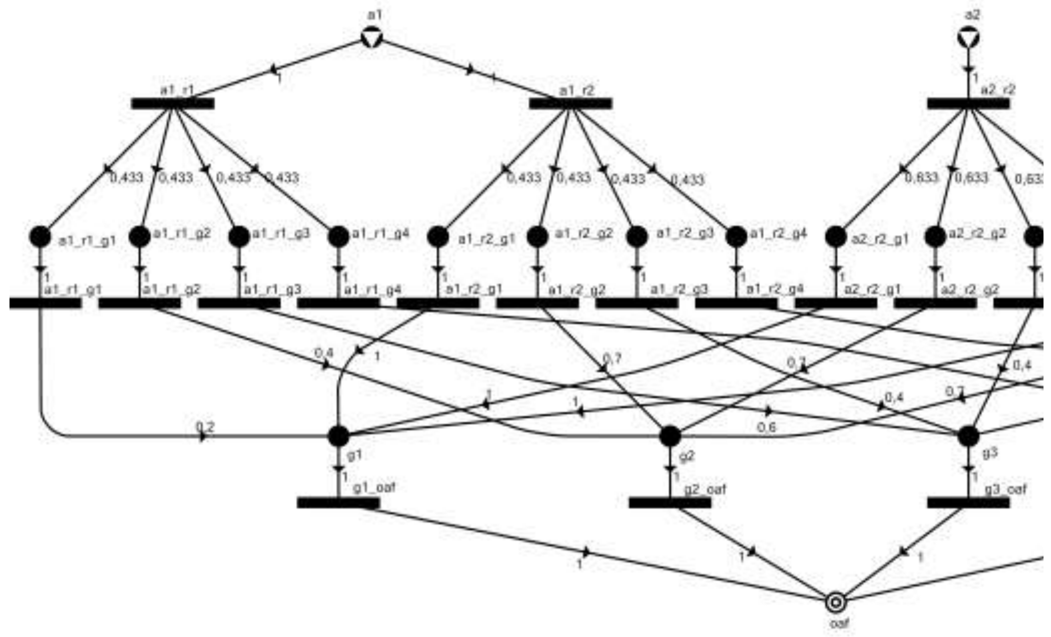


Figure 18. Maximal structure for the hypothetical example to illustrate the solution-structure generation with algorithm MSG.

3.3.3 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 [18,51,83,95,115,116]; 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 dissertation, 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 [115,116].

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 $O = \wp(M) \times \wp(M)$. 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. In addition, lower bound L_{R_j} and upper bound U_{R_j} are specified for each resource R_j. It is important to mention that the cost of each resource R_j, i.e., agent, is not include in the cost function of the model in order to yield a solution equivalent to those in the literature [115,116]. In any case, the total cost of the agents' organization is calculated as follows:

$$organization\_cost = \sum_{R_j \in M \cap R} c_{R_j} \tag{3.1}$$

In addition, lower bound L_{P_j}, upper bound U_{P_j} and its cost c_{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 $y_{O_i} = 1$, 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, $x_{O_i} \leq y_{O_i} U_{O_i}$, where U_{O_i} is the upper limit for the capacity; if such an upper limit does not exist, the U_{O_i} can be any large number L. Finally, z, maximal, is the objective value; representing the oaf function. The resultant $MILP$ model is given in the following.

$$z = max\left(\sum_{P_j \in M \cap P}\left(c_{P_j} * \sum_{O_i \in O} a_{ij} * x_{O_i}\right)\right) \tag{3.2}$$

subject to

$$M = \bigcup_{(\alpha_i,\beta_i) \in O} \alpha_i \cup \beta_i \tag{3.3}$$

$$0 \leq x_{O_i}, L_{O_i} \leq x_{O_i} \leq U_{O_i} \qquad \forall O_i \in O \tag{3.4}$$

$$L_{P_j} \leq \sum_{O_i \in O} a_{ij} * x_{O_i} \leq U_{P_j} \qquad \forall P_j \in M \cap P \tag{3.5}$$

$$L_{R_j} \leq \sum_{O_i \in O} a_{ji} * x_{O_i} \leq U_{R_j} \qquad \forall R_j \in M \cap R \tag{3.6}$$

$$L_{M_j} \leq \sum_{O_i \in O} a_{ji} * x_{O_i} - \sum_{O_i \in O} a_{ij} * x_{O_i} \leq U_{M_j} \qquad \forall M_j \in M \backslash (R \cup P) \tag{3.7}$$

$$x_{O_i} \leq y_{O_i} L \tag{3.8}$$

$$y_{O_i} \in \{0,1\} \tag{3.9}$$

The maximal structure serves as the input to the generation and solution of the MILP model by algorithm ABB [35]. 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[11,12] in less than 75 seconds on an Intel(R) Core(TM) i5 CPU @ 3.20 GHz. Table 8 shows 10 feasible solutions for the example. Algorithms MSG and ABB have been executed by software PNS Studio [88].

3.4 Assessment of Organization based Multi-agent System Design by the Mathematical Programming Model Method

To empirically evaluate the flexibility of the different agent-based organization designs identified by algorithm ABB (see Section 3.3.3), we have developed a simulation that steps through the CRST application. To measure the flexibility, the approach deployed in [95] is followed; specifically, capability failure has been simulated. It is important to mention that capabilities are the key to determining exactly which agents can be assigned to what roles in the organization. Recall that, OMACS defines capabilities as atomic entities used to define the abilities of agents. Thus, capabilities can capture soft abilities such as the ability to access resources, communicate, migrate, or computational algorithms. They also capture hard capabilities such as those of hardware agents such as robots, which include sensors and effectors [95]. 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. The best assignment defines how well an agent, $a_i \in A_{OMACS}$, can play a role,

[11] 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 [18]. For example, assignment set $\phi = \{\langle a_1, r_1, g_1\rangle, \langle a_1, r_1, g_3\rangle, \langle a_2, r_2, g_4\rangle, \langle a_3, r_2, g_4\rangle \}$ is a valid assignment; however it is unfeasible for the motivational example: goal g_2 will never be achieved.

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

$r_k \in R_{OMACS}$, to achieve a goal, $g_j \in G_{OMACS}$ (refer to Eq.1.16). 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.

Table 8. Subset of Feasible Solutions (less than 1%) generated by algorithm $OMACStoPNS$

Sol. #	agent's organization/team assignment set, □	oaf value	Organization's 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
45654	$\left\{\begin{matrix}\langle a_1,r_1,g_1\rangle,\langle a_1,r_1,g_2\rangle,\langle a_1,r_1,g_3\rangle,\langle a_1,r_1,g_4\rangle,\\ \langle a_1,r_2,g_1\rangle,\langle a_1,r_2,g_2\rangle,\langle a_1,r_2,g_3\rangle,\langle a_1,r_2,g_4\rangle\end{matrix}\right\}$	1,8186	850
57730	$\left\{\langle a_2,r_2,g_1\rangle,\langle a_2,r_2,g_2\rangle,\langle a_2,r_2,g_3\rangle,\langle a_2,r_2,g_4\rangle\right\}$	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

Each agent-based organization has been simulated for failure rates ranging from 0 to 100% for 1000 system executions. Comparison of Figure 19 and Figure 20 reveals a difference among the agent-based organization configurations, thereby rendering it possible to offer important remarks about the claim, "the higher the organization score (i.e., the oaf function), the better the performance of the organization."[115]. 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 19 displays a scenario where an agent-based organization, i.e., Sol. # 19883[13], 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[14], with an oaf value of φ= 4,3112 and the cost of $2700. Notice that, the best agent organization is the maximal structure of the PNS problem generated by algorithm $OMACStoPNS$.

Also, Figure 20 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[15] by other agent-based organizations, i.e. Sol. #25400 and Sol. #57730; with oaf values of φ = 2.3452 and φ = 1.3926 ; and, the costs of $1800, and $900; respectively.

[13] media $\dot{x}$ = 63.16 and standard deviation σ=41.84 after 1000 system executions.

[14] media $\dot{x}$ = 64.44 and standard deviation σ=41.35 after 1000 system executions.

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

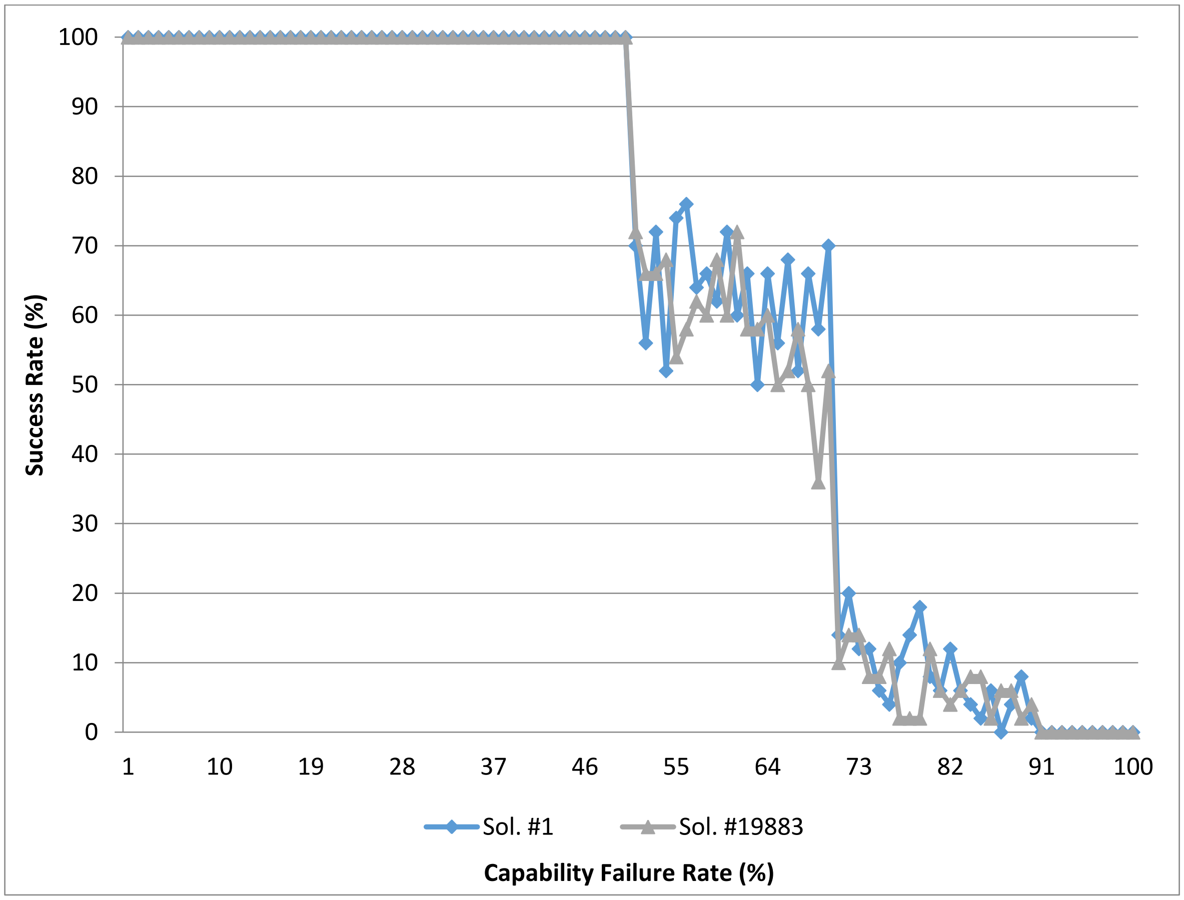


Figure 19. Comparison of Sol. #1 and Sol. # 19883.

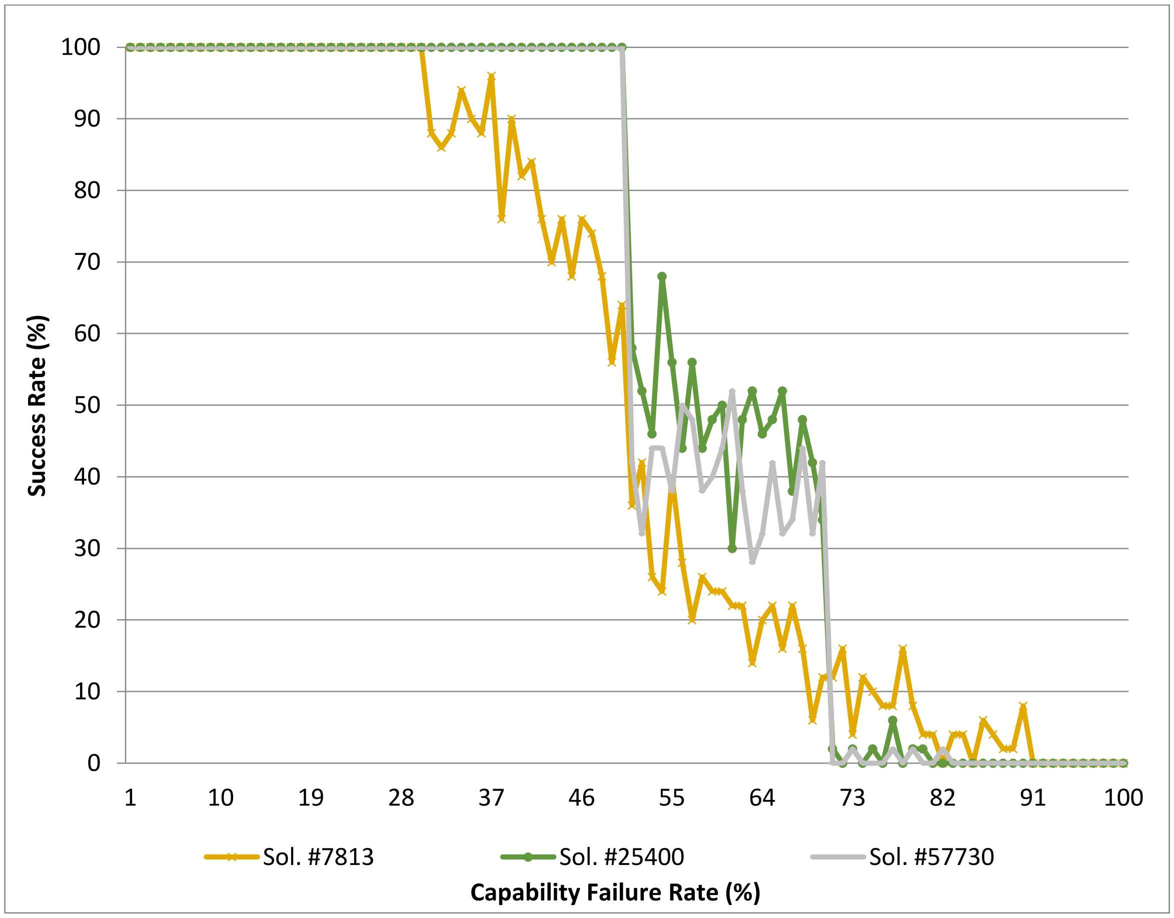


Figure 20. Comparison of Sol. #7813, Sol. #25400, and Sol. #57730.

3.5 Modeling Organization-based Multiagent Systems via Absorbing-Markov Chains

In order to effectively capture the expected behavior of an organizational-based multiagent-system in design phase, a modified version of OMACS is introduced. These modifications allow us to capture the key concepts to modeling the reliability of an organization-based multiagent system under design. As results, two new algorithms are specified: $OMACStoRelaxedPNS$ and $\phi_{OMACS}toR_{\phi_{OMACS}}$. The goals of these two algorithms are: (i) the transformation of an OMACS design model into a relaxed PNS problem, and (ii) the transformation of an organization-based multiagent system assignment set, ϕ_{OMACS}, into an absorbing-markov chain, P_{markov}, and, subsequently, the computation of its steady state, $x_n^{(k)}$ [40].

3.5.1 Modified Version of OMACS

Figure 21 shows the modified, and simplified, version of OMACS. The most significant changes are: (i) $achieves$ function has been modified; (ii) $capable$ function has been removed; and, (iii) functions $possesses$, $potential$, and oaf have been replaced by functions $capReliability$, $asgmtReliability$ and $R_{S_{\Phi_{OMACS}}}$, respectively. These changes are briefly described in what follows.

$achieves$, a function that assumes a role in R_{OMACS}, thereby yielding a set of goals ($achieves$, $R_{OMACS} \rightarrow \wp(G_{OMACS})$, defines the set of goals achieved by that role); $capReliability$, 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] ($capReliability$, $A_{OMACS}\ x\ C_{OMACS} \rightarrow [0,1]$, defines the ability of an agent´s capability to function under stated conditions for a specified period of time, t); $asgmtReliability$, a function whose inputs are an agent in A_{OMACS}, a role in R_{OMACS}, a goal in G_{OMACS} and generates an output, which is a positive real number greater than or equal to 0 and less than or equal to 1 ($asgmtReliability$, $A_{OMACS}\ x\ R_{OMACS}\ x\ G_{OMACS} \rightarrow [0,1]$, defines the reliability of an agent to play a role to achieve a goal), thus giving rise to

$$asgmtReliability(a_i, r_j, g_k) = \begin{cases} 0 & if\ g_k \in achieves(r_j) \\ \prod_{c\ \in\ requires(r_j)} capReliability(a_i, c) & elsewhere; \end{cases} \quad (3.10)$$

Finally, the selection of Φ_{OMACS} from the set of assignments, $asgmtReliability$, is defined by the organization's reorganization function, $R_{S_{\Phi_{OMACS}}}$, that assumes a set of assignments in Φ_{OMACS}, thereby yielding a positive real number in the range of [0,1] ($R_{S_{\Phi_{OMACS}}}$, $\wp(\Phi_{OMACS}) \rightarrow [0,1]$, defines the overall reliability of the agent´s organization in terms of a proposed set of assignments, i.e., Φ_{OMACS}), thus resulting in algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ (seeFigure 26).

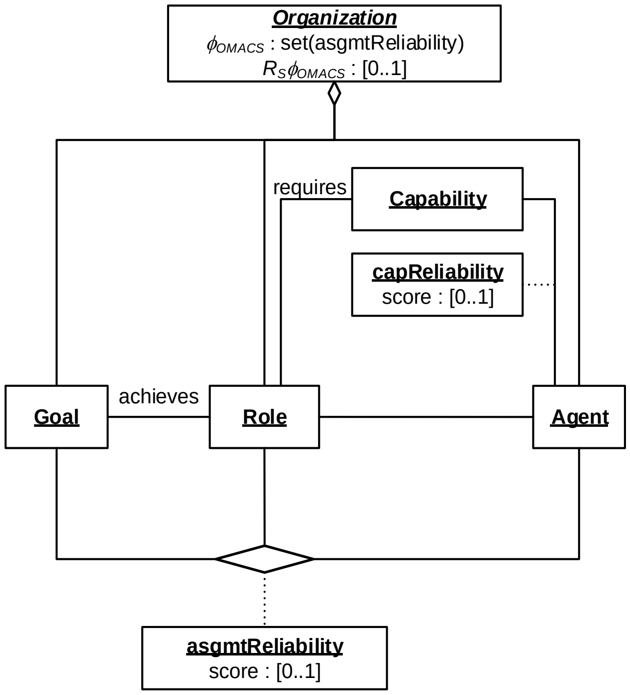


Figure 21. Modified version of the OMACS Meta-model.

By adopting the modified version OMACS, the CRST problem can be represented as shown in Figure 22.

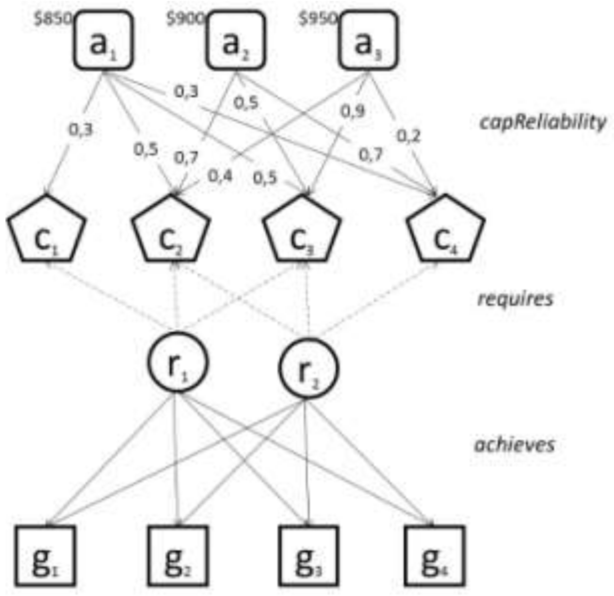


Figure 22. View of the CRST Organization by adopting the modified version of the OMACS meta-model.

3.5.2 Algorithm $OMACStoRelaxedPNS$

The aim of algorithm $\boldsymbol{OMACStoRelaxedPNS}$ is to transform and organization-based multiagent systems design model, following the modified OMACS meta-model previously introduced, into a relaxed PNS problem. By relaxed, we mean, a PNS problem where the constraints are not important as the network model of the PNS problem by itself. Figure 23 shows the different steps of algorithm $\boldsymbol{OMACStoRelaxedPNS}$. A couple of difference can be noted between algorithms $\boldsymbol{OMACStoRelaxedPNS}$ and $\boldsymbol{OMACStoPNS}$ (see Figure 17). The first difference relies on the fact that one operating unit, i.e., $\boldsymbol{system\_goals}$, requires as input all the goals of the systems materials. In this way, unfeasible solutions are not generated by algorithm SSG. The second difference can be clearly seen when the constraint of the different operating units, raw material, and final product are not taken into account in the resultant PNS problem. As result, Figure 24 shows the resultant network structure of the given PNS problem. As mentioned before, the network representation of the PNS problem, by itself, is of our interest for assessing organization-based multiagent system design model.

```
input: $G_{OMACS}, A_{OMACS}, R_{OMACS}, achieves, capReliability, requires$
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 set of goals achieved by a role, $(R_{OMACS} \rightarrow \wp(G_{OMACS}))$, $capReliability$
the ability of an agent´s capability to function under stated conditions for a specified period of time, $t$
$(A_{OMACS} \times C_{OMACS} \rightarrow [0 \dots 1])$, and $requires$ defines the set of capabilities required to play a role
$(R_{OMACS} \rightarrow \wp(C_{OMACS}))$.
output: sets $P, R, O$
comment: $R \subset M, P \subset M, R \cap P = \emptyset$
begin
comment: initialization part of the algorithm;
st1:   $M := M \cup \{overall\_goal\}$;
st2:   $P := P \cup \{overall\_goal\}$;
st3:   $aux := \emptyset$;
lp1:   for $a_i \in A_{OMACS}$ do
       begin
           $R := R \cup \{a_i\}; M := M \cup \{a_i\}$;
       end;
comment: construction part of the algorithm;
lp2:   for $g_j \in G_{OMACS}$ do
       begin
           $M := M \cup \{g_j\}; aux := aux \cup \{g_j\}$;
       end;
st4:  $system\_goals := \{\{aux\}, \{overall\_goal\}\}$; $O := O \cup \{system\_goals\}$;
lp3:   for $a_i \in A_{OMACS}$ do
       begin
           $Capabilities_{a_i} := \emptyset$;
           for $\langle a', c, value' \rangle \in capReliability$ 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 $\wp(c) \subseteq Capabilities_{a_i}$ then
              $aux := \emptyset$;
              for $\langle r'', \wp(g) \rangle \in achieves$ do
              begin
                  for $g_j \in \wp(g)$ do
                  begin
                      if $r_k = r''$ then
                         $M := M \cup \{a_i\_r_k\_g_j\}$; $a_i\_r_k\_g_j := \{\{a_i\_r_k\_g_j\}, \{g_j\}\}$;
                         $aux := aux \cup \{a_i\_r_k\_g_j\}$; $O := O \cup \{a_i\_r_k\_g_j\}$;
                      end;
                      if $aux$ is not empty then
                         $a_i\_r_k := \{\{a_i\}, \{aux\}\}$; $O := O \cup \{a_i\_r_k\}$;
                      end;
                  end;
              end;
       end;
end;
```

Figure 23. Algorithm *OMACStoRelaxedPNS* written in Pidgin Algol (see Appendix C).

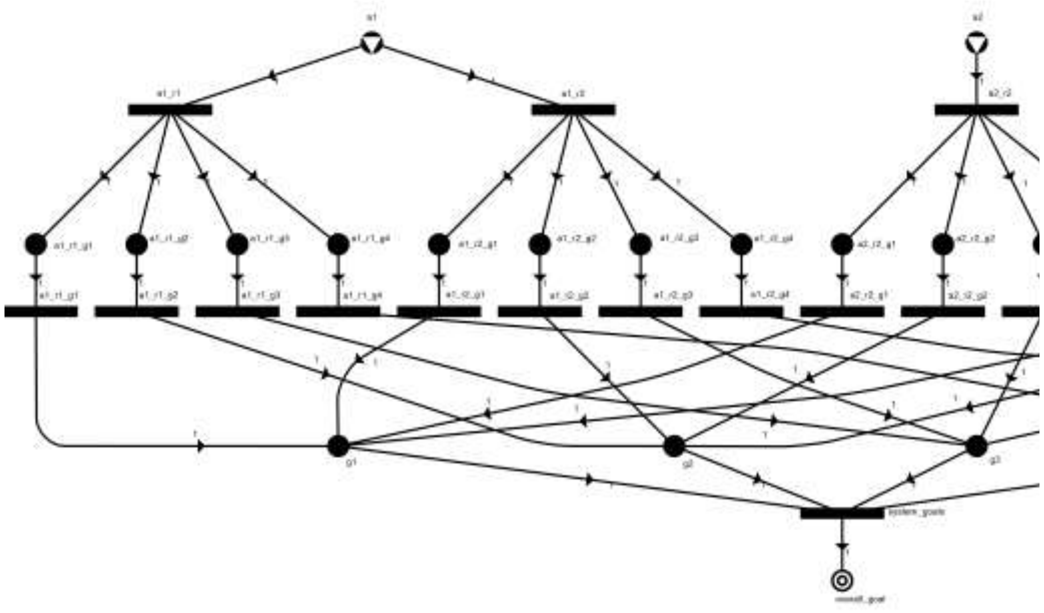


Figure 24. Maximal structure for the hypothetical relaxed example.

3.5.3 Algorithm $\boldsymbol{\phi_{OMACS}toR_{\phi_{OMACS}}}$

The aim of algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ is to transform an organization-based multiagent system assignment set, ϕ_{OMACS}, into an absorbing-markov chain, P_{markov}, and compute its steady state, $x_n^{(k)}$ (refer to Appendix D). That is to say, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ is to evaluate whether or not an assignment set in ϕ_{OMACS} leads the organization-based multiagent system into one of the absorbing states, i.e., either the agents' organization achieve all its goals (success state) or fails to (failure state). Figure 25 shows the steps required for algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ to assess every feasible assignment set in ϕ_{OMACS}.

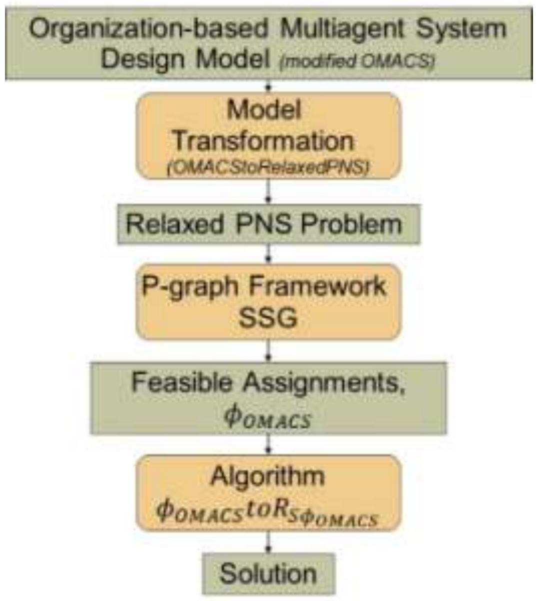

Organization-based Multiagent System Design Model (modified OMACS)
Model Transformation (OMACStoRelaxedPNS)
Relaxed PNS Problem
P-graph Framework SSG
Feasible Assignments, ϕ_{OMACS}
Algorithm $\phi_{OMACS}toR_{S\phi_{OMACS}}$
Solution

Figure 25. Steps required for assessing organization-based multiagent system design model via the algorithm $\boldsymbol{\phi_{OMACS}toR_{\phi_{OMACS}}}$.

Algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$, see Figure 26, comprises three mayor parts, the initialization, the recursion, and the calculation of the steady state, $x_n^{(k)}$, of P_{markov}. The initialization part (statements $st1$, $st2$, $st3$, $st4$, and $st5$) specifies the sets for storing both the absorbing and transient states of P_{markov}. The recursion part (statement $st6$) specifies P_{markov} by describing its state space, S, based upon the assignment set, ϕ_{OMACS}. Finally, the calculation part (statements $st7$, $st8$, $st9$, $st10$, $st11$, $st12$, $st13$, $st14$, and loop $loop1$) computes $x_n^{(k)}$ (refer to Figure D.3 in Appendix D). In what follows, a couple of iterations of algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ are presented as illustration.

Initially, variable n is assigned the integer value 1. n keeps track of the total number of states in S. Subsequently, sets S_S, S_F, S_T, and S are assigned the empty set. That is, $S_S = \emptyset$, $S_F = \emptyset$, $S_T = \emptyset$, and $S = \emptyset$. In that order, S_S stores pairs of the form $\langle i, p_{i\,success}\rangle$, where $p_{i\,success}$ symbolizes the transition probability from state i to state $success$. State

$success$ symbolizes the accomplishments of the multiagent systems[16]. S_F collects pairs of the form $\langle i, p_{i\,failure}\rangle$, where $p_{i\,failure}$ denotes the transition probability from state i to state $failure$. State $failure$ represents failure of the organization-based multiagent systems[17]. S_T stores pairs of the form $\langle j, \langle i, p_{ij}\rangle\rangle$, where p_{ij} means the transition probability between transient states i to j; and, S collects pairs of the form $\langle i, label_i\rangle$, where $label_i$ specifies the name of state i in S of P_{markov}. Valid state names can be: $success$, $failure$, or $a_1 \cdot g_1$. Where state $a_1 \cdot g_1$ symbolizes agent a_1 is still available in the system and goal g_1 must be accomplished. Subsequently, recursive procedure $AMC - Spec$ is invoked. The outcome of this recursive procedure is an absorbing markov chain, i.e., P_{markov} (see Figure 27).

Depending on the cardinality of A_{OMACS} and G_{OMACS}, procedure $AMC - Spec$ evaluates three cases (see Figure 27): First, if $|A_{OMACS}| = 1$ and $|G_{OMACS}| > 1$, the transition probability is calculated as the system reliability of a simple series system (see Appendix E). That is to say, the reliability that one agent, in A_{OMACS}, achieves the entire set of goals, in G_{OMACS}, thru a set of roles, in R_{OMACS}, is equivalent to the product of the best assignment $\langle a_i, r_j, g_k\rangle$ for each $a_i \in A_{OMACS}$, $r_j \in R_{OMACS}$, and $g_k \in G_{OMACS}$, where $|A_{OMACS}| = 1$, $|R_{OMACS}| \geq 1$, and $|G_{OMACS}| > 1$. Hence, the basic equation for this case is:

$$R_{S_{\phi_{assignments}}} = asgmtReliability_{(a_1, r_j, g_1)} \times asgmtReliability_{(a_1, r_j, g_2)} \times \cdots \times asgmtReliability_{(a_1, r_j, g_k)} \tag{3.11}$$

[16] In other words, the agents' organization is able to achieve its goals.

[17] In other words, the agents' organization is not able to achieve its goals.

Second, if $|A_{OMACS}| > 1$ and $|G_{OMACS}| = 1$, the transition probability is calculated as the system reliability of a simple parallel system (see Appendix E). Particularly, the reliability that more than one agent, in A_{OMACS}, achieves one goal, in G_{OMACS}, thru a set of roles, in R_{OMACS}, is equivalent to the product of the best assignment $\langle a_i, r_j, g_k \rangle$ for each $a_i \in A_{OMACS}$, $r_j \in R_{OMACS}$, and $g_k \in G_{OMACS}$, where $|A_{OMACS}| > 1, |R_{OMACS}| \geq 1$, and $|G_{OMACS}| = 1$. Thus, the basic equation for this case is:

$$R_{S_{\phi_{assignments}}} = 1 - \begin{pmatrix} \left(1 - asgmtReliability_{(a_1, r_j, g_1)}\right) \times \\ \left(1 - asgmtReliability_{(a_2, r_j, g_1)}\right) \times \\ \cdots \times \\ \left(1 - asgmtReliability_{(a_i, r_j, g_1)}\right) \end{pmatrix} \tag{3.12}$$

and, the probability of failure is defined as follows

$$R_{F_{\phi_{assignments}}} = 1 - R_{S_{\phi_{assignments}}} \tag{3.13}$$

Finally, if both $|A_{OMACS}| > 1$ and $|G_{OMACS}| > 1$, the system is considered neither series nor parallel. Therefore, the transition probability is calculated as the product of a finite set of mutually independent events, where each event can be either an agent achieving a given goal or failing to accomplish it. Notice that, first case 1, i.e., $|A_{OMACS}| = 1$ and $|G_{OMACS}| > 1$, and second case, i.e., $|A_{OMACS}| > 1$ and $|G_{OMACS}| = 1$, are the base cases of this sub-procedure; while third case, i.e., $|A_{OMACS}| > 1$ and $|G_{OMACS}| > 1$, is the recursive case.

For the sake of the hypothetical example, since the CRST system is composed of three agents, i.e., $|A_{OMACS}| = 3$, whose aim is to accomplish four goals, i.e., $|G_{OMACS}| = 4$, sub-procedure $NonSerParalSyst$ is first called[18] (Figure 28). Consequently, an optimal assignment set, $\phi_{assignments}$, is generated by invoking procedure $optimalAssignment$ (Figure 29). $\phi_{assignments}$ contains the best possible assignments $\langle a_i, r_k, g_j \rangle$ for the given

[18] First invocation of sub-procedure *NonSerParalSyst*.

sets A_{OMACS}, R_{OMACS}, and G_{OMACS}. For computing $\phi_{assignments}$, procedure $optimalAssignment$ evaluates for each $a_i \in A_{OMACS}$ and $g_j \in G_{OMACS}$ the best $r_k \in R_{OMACS}$ by calculating the assignment reliability of a_i, r_k, and g_j (see Eq. 3.10).

```
input: $A_{OMACS}, G_{OMACS}$
comment: $A_{OMACS}$ a set of agents, $G_{OMACS}$ a set of Goals
output: the reliability, $R_{S_{\phi_{OMACS}}}$ , of $\phi_{OMACS}$
global variables: $S_S, S_F, S_T, S, n, achieves, possesses, requires, R_{OMACS}, \phi_{OMACS}$
comment: $S$ is the state space of $P_{markov}$ , set $S_S$ stores elements of the form
$\langle i, p_{i\ success} \rangle$, where $p_{i\ success}$ represents the transition probability from state i to
state $success$ set $S_F$ stores elements of the form $\langle i, p_{i\ failure} \rangle$, where $p_{i\ failure}$ represents
the transition probability from state i to state $failure$; and; $S_T$ stores elements of the
form $\langle j, \langle i, p_{ij} \rangle \rangle$ , where $p_{ij}$ represents the transition probability between transient
states $i$ to $j$. $achieves$ assumes a role in $R_{OMACS}$ yielding a set of goals $(achieves, R_{OMACS} -$
$\wp(G_{OMACS}))$, $requires$ defines the set of capabilities required to play a role $(R_{OMACS} \rightarrow$
$\wp(C_{OMACS}))$, $capReliability$ defines the ability of an agent´s capability to function under
stated conditions for a specified period of time, $t$ $(capReliability, A_{OMACS}\ x\ C_{OMACS} \rightarrow$
$[0,1])$; $R_{OMACS}$ a set of roles, $\phi_{OMACS}$ defines the set of agent – role – goal tuples
$\langle a_i, r_j, g_k \rangle$, indicating that agent $a_i \in A_{OMACS}$ has been assigned to play role $r_j \in R_{OMACS}$
in order to achieve goal $g_k \in G_{OMACS}$ ($\phi_{OMACS}$ is a subset of all the potential assignments
of agents to play roles to achieve goals);

begin
st1:  $n := 1$;
st2:  $S_S := \emptyset$;
st3:  $S_F := \emptyset$;
st4:  $S_T := \emptyset$;
st5:  $S := \emptyset$;
st6:  $\boldsymbol{AMC-Spec}(A_{OMACS}, G_{OMACS}, n-1)$;
st7:  $S_S := S_S \cup \{(n,1)\}$;
st8:  $S_S := S_S \cup \{(n+1,0)\}$;
st9:  $S_F := S_F \cup \{(n,0)\}$;
st10:  $S_F := S_F \cup \{(n+1,1)\}$;
st11:  $x[1{:}n+2]$;
st12:  $x[1] = 1$;
st13:  $\delta := 1 - x[n+1]$;
st14:  $\varepsilon = 0.000001$;
loop1:  while $\delta > \varepsilon$ do
          begin
                 $v = x[n+1]$;
                 $x := \boldsymbol{matrixProduct}(S_T, S_S, S_F, x)$;
          $error := \|x[n+1] - v\|$;
          end;
st15:  $R_{S_{\phi_{OMACS}}} := x[n+1]$;
end;
```

Figure 26. Algorithm $\boldsymbol{\phi_{OMACS} toR_{\phi_{OMACS}}}$ written in Pidgin (see Appendix C).

```
comment: $parent$ captures the current state $i$, in $S$, such $p_{ij}$, probability transition, from
state $i$ to state $j$ can be computed
Procedure $AMC - Spec(A_{OMACS}, G_{OMACS}, parent)$:
begin
st1:    if $|A_{OMACS}| = 1$ & $|G_{OMACS}| \geq 1$ then
        begin
            go to $SeriesSystem$;
        end;
        else if $|A_{OMACS}| > 1$ & $|G_{OMACS}| = 1$ then
        begin
            go to $ParallelSystem$;
        end;
        else
        begin
            go to $NonSerParalSyst$;
        end;
end;
```

Figure 27. Procedure *AMC-Spec*.

As result, matrix $M_{\phi_{assignments}}$ is created

$$M_{\phi_{assignments}} = \begin{bmatrix} \langle\{a_1, r_1, g_1\}, 0.075\rangle & \langle\{a_1, r_1, g_2\}, 0.075\rangle & \langle\{a_1, r_1, g_3\}, 0.075\rangle & \langle\{a_1, r_1, g_4\}, 0.075\rangle \\ \langle\{a_2, r_2, g_1\}, 0.245\rangle & \langle\{a_2, r_2, g_2\}, 0.245\rangle & \langle\{a_2, r_2, g_3\}, 0.245\rangle & \langle\{a_2, r_2, g_4\}, 0.245\rangle \\ \langle\{a_3, r_2, g_1\}, 0.072\rangle & \langle\{a_3, r_2, g_2\}, 0.072\rangle & \langle\{a_3, r_2, g_3\}, 0.072\rangle & \langle\{a_3, r_2, g_4\}, 0.072\rangle \end{bmatrix}$$

It can be noted that, the structure of matrix $M_{\phi_{assignments}}$ is given by

$$M_{\phi_{assignments}} = \begin{bmatrix} \langle\{a_1, r_j, g_1\}, R_{(a_1, r_j, g_1)}\rangle & \cdots & \langle\{a_1, r_j, g_k\}, R_{(a_1, r_j, g_k)}\rangle \\ \vdots & \ddots & \vdots \\ \langle\{a_i, r_j, g_1\}, R_{(a_i, r_j, g_1)}\rangle & \dots & \langle\{a_i, r_j, g_k\}, R_{(a_i, r_j, g_k)}\rangle \end{bmatrix}, |A_{OMACS}| \geq |G_{OMACS}|$$

$$M_{\phi_{assignments}} = \begin{bmatrix} \langle\{a_1, r_j, g_1\}, R_{(a_1, r_j, g_1)}\rangle & \cdots & \langle\{a_i, r_j, g_1\}, R_{(a_i, r_j, g_1)}\rangle \\ \vdots & \ddots & \vdots \\ \langle\{a_1, r_j, g_k\}, R_{(a_1, r_j, g_k)}\rangle & \dots & \langle\{a_i, r_j, g_k\}, R_{(a_i, r_j, g_k)}\rangle \end{bmatrix}, |A_{OMACS}| < |G_{OMACS}|$$

```
$\boldsymbol{NonSerParalSyst}$   $R_{S_{\phi_{assignments}}} := 1.0;$
                  $\phi_{assignments} := \boldsymbol{optimalAssignments}(A_{OMACS}, G_{OMACS});$
                  $A_{OMACS_{available}} := \wp(A_{OMACS});$
                  for $A_{OMACS_{succeed}} \in A_{OMACS_{available}}$ do
                  begin
                      $A_{OMACS_{copy}} := A_{OMACS};$
                      $G_{OMACS_{copy}} := G_{OMACS};$
                      $A_{OMACS_{failed}} := A_{OMACS} \backslash A_{OMACS_{succeed}};$
                      for $a_s \in A_{OMACS_{succeed}}$ do
                      begin
                              for $\langle a_s^{'}, r_j, g_k, value\rangle \in \phi_{assignments}$ do
                              begin
                                 if $(a_s = a_s^{'})$then
                                 begin
                                         $G_{OMACS_{copy}} := G_{OMACS} - \{g_k\};$
                                         $R_{S_{\phi_{assignments}}} := R_{S_{\phi_{assignments}}} * value;$
                                 end;

                              end;
                      end;
                      for $a_f \in A_{OMACS_{failed}}$ do
                      begin
                              for $\langle a_f^{'}, r_j, g_k, value\rangle \in \phi_{assignments}$ do
                              begin
                                 if $(a_f = a_f^{'})$then
                                 begin
                                         $A_{OMACS_{copy}} := A_{OMACS} - \{a_f\};$
                                         $R_{S_{\phi_{assignments}}} := R_{S_{\phi_{assignments}}} * (1 - value);$
                                 end;
                              end;
                      end;
                  if $\left(|A_{OMACS_{copy}}| > 0 \land |G_{OMACS_{copy}}| = 0\right)$ then
                  begin $S_S = S_S \cup \langle parent, R_{S_{\phi_{assignments}}}\rangle;$ end;
                  if $\left(|A_{OMACS_{copy}}| = 0 \land |G_{OMACS_{copy}}| > 0 \lor |\phi_{assignments}| = 0\right)$ then
                  begin $S_F = S_F \cup \langle parent, R_{S_{\phi_{assignments}}}\rangle;$ end;
                     if $\left(|A_{OMACS_{copy}}| > 0 \land |G_{OMACS_{copy}}| > 0 \land |\phi_{assignments}| > 0\right)$ then
                  begin
                      if $\langle i, \{A_{OMACS_{copy}} \cup G_{OMACS_{copy}}\}\rangle \in S$ then
                      begin $S_T = S_T \cup \langle i, \gamma \cup \{parent, R_{S_{\phi_{assignments}}}\}\rangle$ end;
                      else begin
                              $S_T = S_T \cup \langle n, \gamma \cup \{parent, R_{S_{\phi_{assignments}}}\}\rangle;$
                              $S = S \cup \langle n, \{A_{OMACS_{copy}} \cup G_{OMACS_{copy}}\}\rangle;$
                              $n = n + 1;$
                              $\boldsymbol{AMC - Spec}(A_{OMACS_{copy}}, G_{OMACS_{copy}}, n - 1);$
                           end;
                  end;
```

Figure 28. Procedure *NonSerParalSyst* written in Pidgin (see Appendix C).

Thereafter, the Hungarian method [68,69,79] is called with matrix $M_{\phi_{assignments}}$ as input. The purpose of the Hungarian method in our approach is to guarantee, at each step, the selection of the best assignment of a set of agents to a set of goals thru a set of roles. Accordingly, an optimal assignment or minimum matching, $\phi_{assignments}$, is obtained. That is, $\phi_{assignments} = \{\langle\{a_1, r_1, g_1\}, 0.075\rangle, \langle\{a_3, r_2, g_2\}, 0.0072\rangle, \langle\{a_2, r_2, g_3\}, 0.245\rangle\}$. This minimum matching indicates that agent a_1 should play role r_1 in order to achieve goal g_1 with a probability of success, i.e., reliability, of 0.075; agent a_2 should play role r_2 in order to achieve goal g_3 with a reliability of 0.245; and, agent a_3 should play role r_2 in order to achieve goal g_2 with a reliability of 0.0072. Afterwards, sub-procedure $NonSerParalSyst$ creates set $A_{OMACS_{available}}$, which represents all subsets of A_{OMACS}; such that, every element in $A_{OMACS_{available}}$ symbolizes the combination of agents who are still operative, i.e., are not broken and not undergoing repair, in order to achieve the goals of the system. Thus, $A_{OMACS_{available}}=\{\{a_1, a_2, a_3\}, \{a_1, a_2\}, \{a_2, a_3\}, \{a_1, a_3\}, \{a_1\}, \{a_2\}, \{a_3\}, \emptyset\}$[19].

Afterwards, for each set in $A_{OMACS_{available}}$, copies of A_{OMACS} and G_{OMACS} are created, i.e., $A_{OMACS_{copy}}$ and $G_{OMACS_{copy}}$, respectively. Set $A_{OMACS_{copy}}$ stores the group of agents who do not fail to achieve their goals, and set $G_{OMACS_{copy}}$ stores the group of goals which are still available to be accomplished. Assuming the ordering of $A_{OMACS_{available}}$ presented above, where $\{a_1, a_2, a_3\}$ is the first combination of agents who succeed in achieving their goals, the reliability of the optimal assignment, $R_{S_{\phi_{assignments}}}$ is computed. Because every agent in $\{a_1, a_2, a_3\}$ is assigned a goal in $\phi_{assignments}$[20], $R_{S_{\phi_{assignments}}} = asgmtReliability_{(a_1, r_1, g_1)} * asgmtReliability_{(a_3, r_2, g_2)} * asgmtReliability_{(a_2, r_2, g_3)}$. Hence, $R_{S_{\phi_{assignments}}} = 0.075 * 0.072 * 0.245$, which gives rise to $R_{S_{\phi_{assignments}}} = 0.001323$. Since $A_{OMACS_{copy}} \setminus \{a_1, a_2, a_3\} = \emptyset$, it implies that goals g_1, g_2, and g_3 are achieved; therefore removed from $G_{OMACS_{copy}}$. Afterwards, sub-procedure

[19] Notice that, eight different state transitions are to be evaluated (see Figure 40).

[20] Recall that at this point of the computation $\phi_{assignments} = \{\langle\{a_1, r_1, g_1\}, 0.075\rangle, \langle\{a_3, r_2, g_2\}, 0.0072\rangle, \langle\{a_2, r_2, g_3\}, 0.245\rangle\}$

$NonSerParalSyst$ based upon the size of sets $A_{OMACS_{copy}}$, $G_{OMACS_{copy}}$, and $\phi_{assignments}$, checks if the system has reached either a success state, a failure state, or a transient state.

```
output: an optimal assignment set of $\phi_{OMACS}$, i. e., $\phi_{assignments}$
Procedure optimalAssignment($A_{OMACS}$, $G_{OMACS}$):
begin
st1:  $i := 1$;
st2:  $j := 1$;
st3:  $M_{\phi_{assignments}}$ [1: |$A_{OMACS}$|][1: |$G_{OMACS}$|];
lp1:  for $a_i \in A_{OMACS}$ do
      begin
          for $g_k \in G_{OMACS}$ do
          begin
              $role := \emptyset$;
              $MAX := -\infty$;
              for $r_j \in R_{OMACS}$ do
              begin
                  for $\langle r_j^{'}, g_k^{'} \rangle \in achieves$ do
                  begin
                      if $r_j = r_j^{'}$ & $g_k = g_k^{'}$ then
                      begin
                          $reliability := 1.0$;
                          if $\langle a_i, r_j, g_k \rangle \in \phi_{OMACS}$ then
                          begin
                              $reliability :=$ asgmtReliability($a_i, r_j, g_k$);
                          end;
                          if $reliability > MAX$ then
                          begin
                              $MAX = realibility$;
                              $role = \{r_j\}$;
                          end;
                      end;
                  end;
              end;
              if $role = \emptyset$ then
              begin $M_{\phi_{assignments}}$ $[i][j] = \langle a_i, role, g_k, -\infty \rangle$;
              end;
              else
              begin $M_{\phi_{assignments}}$ $[i][j] = \langle a_i, role, g_k, MAX \rangle$;
              end;
          $j := j + 1$;
          end;
      $j := 1$; $i := i + 1$;
      end;
      $\phi_{assignmens}$ := HungarianAlgorithm($M_{\phi_{assignments}}$ );
end;
```

Figure 29. Procedure *optimalAssignment* written in Pidgin (see Appendix C).

Since $\left|A_{OMACS_{copy}}\right| > 1$, i.e., $A_{OMACS_{copy}} = \{a_1, a_2, a_3\}$, and $\left|G_{OMACS_{copy}}\right| = 1$, i.e., $G_{OMACS_{copy}} = \{g_4\}$, and $|\phi_{assignments}| > 1$, a new transient state of the system is created, i.e., $a_1 \cdot a_2 \cdot g_3 \cdot g_4$., and its corresponding transition probability is to be updated[21].

[21] The system either has not succeed or failed to achieve its given set of goals.

Consequently, state $\{a_1 \cdot a_2 \cdot a_3 \cdot g_4\}$ is added to set S. Because $\{a_1 \cdot a_2 \cdot a_3 \cdot g_4\} \notin S$, sub-procedure $NonSerParalSyst$ updates sets S and S_T and variable n. Thus, $S = S \cup \langle 1, a_1 \cdot a_2 \cdot a_3 \cdot g_4 \rangle$, $S_T = S_T \cup \langle 1, \{0,0.001323\} \rangle$ and $n = 2$.

Subsequently, the recursive procedure $AMC - Spec$ is called again22, with parameter $A_{OMACS} = \{a_1, a_2, a_3\}$, $G_{OMACS} = \{g_4\}$, and $parent = (n-1)$, i.e., $parent = 1$. Since $|A_{OMACS}| > 1$ and $|G_{OMACS}| = 1$, sub-procedure $ParalSyst$ (see Figure 30) is invoked and the transition probability is calculated as a simple parallel system where two cases are to be evaluated: success and failure. Hence, by Eq. 3.12, $R_{S_{\phi_{assignment}}} = (1 - ((1-0.075) * (1-0.072) * (1-0.245)))$, which gives rise to $R_{S_{\phi_{assignment}}} = 0.648092$. Also, by Eq.3.13, $R_{F_{\phi_{assignments}}} = 1 - 0.648092$, which turns out in $R_{F_{\phi_{assignments}}} = 0.351908$. Hence, sets S_S and S_F are updated to $S_S = S_S \cup \{\langle 1,0.648092 \rangle\}$ and $S_F = S_F \cup \{\langle 1,0.351908 \rangle\}$ (see Figure 31).

[22] Second recursive call to this procedure.

```
$\boldsymbol{ParalSyst}$   $reliability \coloneqq 1.0;$
             for $g_k \in G_{OMACS}$ do
             begin
                 $MAX = \infty$;
                 for $\langle r_j, g_k' \rangle \in achieves$ do
                 begin
                     $RsParalSyst \coloneqq 1.0;$
                     if $g_k = g_k'$ then
                     begin
                         for $a_i \in A_{OMACS}$ do
                         begin
                             if $\langle a_i, r_j, g_k \rangle \in \phi_{assignments}$ then
                             begin
                                 $RsParalSyst \coloneqq asgmtReliability(a_i, r_j, g_k);$
                             end;
                             else
                             begin
                                 $RsParalSyst \coloneqq 0.0;$
                             end;
                         end;
                     end;
                     if $RsParalSyst > MAX$ then
                     begin
                         $MAX \coloneqq RsParalSyst;$
                     end;
                 end;
                 $reliability \coloneqq reliability * MAX;$
             end;
             $reliability \coloneqq 1.0 - reliability;$
             $S_S \coloneqq S_S \cup \langle parent, reliability \rangle;$
             $S_F \coloneqq S_F \cup \langle parent, 1.0 - reliability \rangle;$
```

Figure 30. Procedure $\boldsymbol{ParalSyst}$ written in Pidgin (see Appendix C).

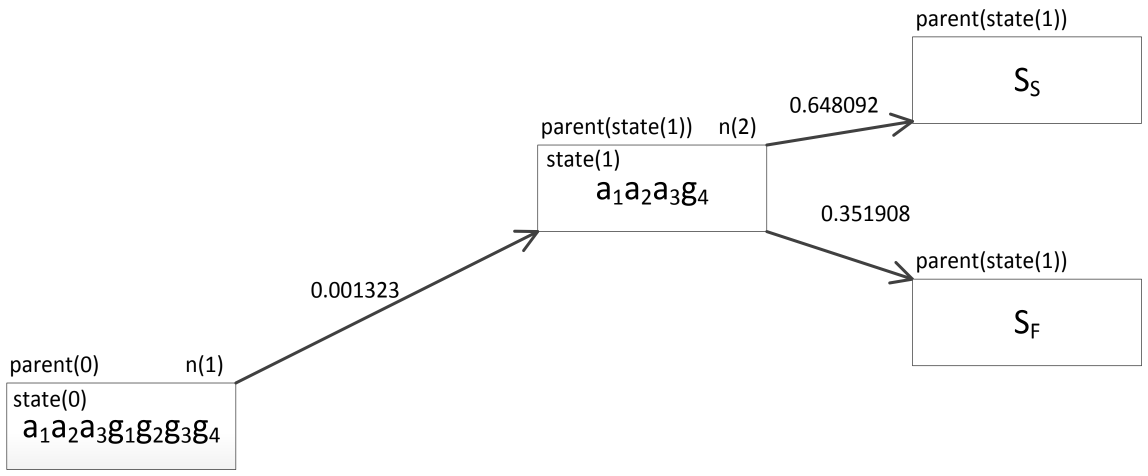


Figure 31. Branching of State 1, i.e., $\boldsymbol{a_1 \cdot a_2 \cdot g_3 \cdot g_4}$

Consequently, the next element in $A_{OMACS_{available}}$ is evaluated, i.e., $\{a_1, a_2\}$[23]. As result, the reliability of the optimal assignment, $R_{S_{\phi_{assignments}}}$ is computed. Because $a_3 \notin \{a_1, a_2\}$,
$R_{S_{\phi_{assignments}}} = asgmtReliability(a_1, r_1, g_1) \ * \ \big(1 - asgmtReliability(a_3, r_2, g_2)\big) *$ $asgmtReliability(a_2, r_2, g_3)$. Hence, $R_{S_{\phi_{assignments}}} = 0.075 * 0.928 * 0.245$, which gives rise to $R_{S_{\phi_{assignments}}} = 0.017052$. Because $A_{OMACS_{copy}} \backslash \{a_1, a_2\} = a_3$, it implies two things: first a_3 failed to accomplished goal g_2 thru role r_2, therefore removed from $A_{OMACS_{copy}}$, i.e.., $A_{OMACS_{copy}} = \{a_1, a_2\}$; and second, goals g_1 and g_3 are achieved; thus removed from $G_{OMACS_{copy}}$, i.e., $G_{OMACS_{copy}} = \{g_2, g_4\}$. Since $\left|A_{OMACS_{copy}}\right| > 1$, i.e., $A_{OMACS_{copy}} = \{a_1, a_2\}$, and $\left|G_{OMACS_{copy}}\right| > 1$, i.e., $G_{OMACS_{copy}} = \{g_2, g_4\}$, a new transient state of the system is created, i.e., $a_1 \cdot a_2 \cdot g_3 \cdot g_4$. Because $\{a_1 \cdot a_2 \cdot g_3 \cdot g_4\} \notin S,$ sets S and S_T are updated to $S = S \cup \{\langle 1, \{a_1 \cdot a_2 \cdot a_3 \cdot g_4\}\}\rangle, \langle 2, \{a_1 \cdot a_2 \cdot g_3 \cdot g_4\}\}\rangle\}$ and $S_T = \ S_T \cup \{\langle 1, \{0, 0.001323\}\rangle, \ \langle 2, \{0, 0.017052\}\rangle\}$. Additionally, variable n is updated to 3, i.e., $n = 3$.

Accordingly, the procedure $AMC - Spec$ is invoked for second time with parameter $A_{OMACS} = \{a_1, a_2\}$, $G_{OMACS} = \{g_2, g_4\}$, and $parent \ = (n-1)$, i.e., $parent \ = 2$. Notice that, the cardinality of A_{OMACS} and G_{OMACS} is greater than 1; therefore, sub-procedure $NonSerParalSyst$ is invoked for second time (refer to Figure 28). Consequently, an optimal assignment set, $\phi_{assignments}$, is generated for sets A_{OMACS} and G_{OMACS}. As a result, matrix $M_{\phi_{assignments}}$ is created

$$M_{\phi_{assignments}} = \begin{bmatrix} \langle\{a_1, r_1, g_2\}, 0.075\rangle & \langle\{a_1, r_1, g_4\}, 0.075\rangle \\ \langle\{a_2, r_2, g_2\}, 0.245\rangle & \langle\{a_2, r_2, g_4\}, 0.245\rangle \end{bmatrix}.$$

[23] The content of set $A_{OMACS_{assignment}}$ after the first invocation of sub-procedure *NonSerParalSyst* is $A_{OMACS_{assignment}}$=$\{\{a_1, a_2, a_3\}, \ \{a_1, a_2\}, \ \{a_2, a_3\}, \ \{a_1, a_3\}, \{a_1\}, \{a_2\}, \{a_3\}, \emptyset\}$.

Consequently, after invoking the Hungarian method, an optimal assignment or minimum matching, $\phi_{assignments}$, is obtained. That is, $\phi_{assignments} = \{\langle\{a_1, r_1, g_2\}, 0.075\rangle, \langle\{a_2, r_2, g_4\}, 0.245\rangle\}$. Afterwards, sub-procedure $NonSerParalSyst$ creates set $A_{OMACS_{available}}$ (see Figure 28). Thus, $A_{OMACS_{available}} = \{\{a_1, a_2\}, \{a_1\}, \{a_2\}, \{a_3\}, \emptyset\}$[24]. Consequently, for each set in $A_{OMACS_{available}}$, copies of A_{OMACS} and G_{OMACS} are created, i.e., $A_{OMACS_{copy}}$ and $G_{OMACS_{copy}}$, respectively. Assuming the ordering of $A_{OMACS_{available}}$ presented above, where $\{a_1, a_2\}$ is the first combination of agents, the reliability of the optimal assignment, $R_{S_{\phi_{assignments}}}$ is computed. Since every agent in $\{a_1, a_2\}$ is assigned a goal in $\phi_{assignments}$[25], $R_{S_{\phi_{assignments}}} = asgmtReliability(a_1, r_1, g_2) * asgmtReliability(a_2, r_2, g_4)$. Hence, $R_{S_{\phi_{assignments}}} = 0.075 * 0.245$, which gives rise to $R_{S_{\phi_{assignments}}} = 0.018375$. Because $A_{OMACS_{copy}} \setminus \{a_1, a_2\} = \emptyset$, it implies that goals g_2 and g_4 are achieved; therefore removed from $G_{OMACS_{copy}}$. Thus, $G_{OMACS_{copy}} = \emptyset$. Since, $G_{OMACS_{copy}}$ is empty, this implies our system has reached a success state, therefore set S_S is updated to $S_S = S_S \cup \{\langle 1, 0.648092\rangle, \langle 2, 0.018375\}\rangle\}$. Aftrwards, the next element in $A_{OMACS_{available}}$ is evaluated, i.e., $\{a_1\}$. As result, the reliability of the optimal assignment, $R_{S_{\phi_{assignments}}}$ is computed. Because $a_2 \notin \{a_1\}$, $R_{S_{\phi_{assignments}}} = asgmtReliability(a_1, r_1, g_3) * (1 - asgmtReliability(a_2, r_2, g_4))$. Hence, $R_{S_{\phi_{assignments}}} = 0.075 * 0.755$, which gives rise to $R_{S_{\phi_{assignments}}} = 0.056625$. Since $A_{OMACS_{copy}} \setminus \{a_1\} = a_2$, it implies two things: first a_2 failed to accomplished goal g_4 thru role r_2, therefore removed from $A_{OMACS_{copy}}$, i.e.., $A_{OMACS_{copy}} = \{a_1\}$; and second, goal g_3 is achieved; thus removed from $G_{OMACS_{copy}}$, i.e., $G_{OMACS_{copy}} = \{g_4\}$. Since $\left|A_{OMACS_{copy}}\right| > 1$, i.e., $A_{OMACS_{copy}} = \{a_1\}$, and $\left|G_{OMACS_{copy}}\right| > 1$, i.e., $G_{OMACS_{copy}} = \{g_4\}$, a new transient state of the system is created, i.e., $a_1 \cdot g_4$. Because $\{a_1 \cdot g_4\} \notin S$, sets S and S_T are updated to $S = S \cup \{\langle 1, a_1 \cdot a_2 \cdot a_3 \cdot$

[24] Notice that, four different state transitions are to be evaluated (see Figure 40).

[25] Recall that at this point of the computation, second call of procedure *AMC-Spec*, $\phi_{assignments} = \{\langle\{a_1, r_1, g_2\}, 0.075\rangle, \langle\{a_2, r_2, g_4\}, 0.245\rangle\}$.

$g_4\rangle, \langle 2, a_1 \cdot a_2 \cdot g_3 \cdot g_4\rangle, \langle 3, a_1 \cdot g_4\rangle\}$ and $S_T = S_T \cup \{\langle 1, \{0{,}0.001323\}\rangle,\ \langle 2, \{0{,}0.017052\}\rangle,\ \langle\{3, \{2{,}0.056625\}\}\rangle\}$. Additionally, variable n is updated to 4, i.e., $n = 4$.

Subsequently, the procedure $AMC - Spec$ is invoked for third time with parameters $A_{OMACS} = \{a_1\}$, $G_{OMACS} = \{g_4\}$, and $parent = (n-1)$, i.e., $parent = 3$. Since $|A_{OMACS}| = 1$ and $|G_{OMACS}| = 1$, sub-procedure $SerSyst$ is invoked and the transition probability is calculated as a simple series system where two cases are to be evaluated: success and failure. Hence, by Eq. 3.11, $R_{S_{\Phi_{assignments}}} = 0.075$. Also, by Eq. 3.13, $R_{F_{\Phi_{assignments}}} = 1 - 0.075$, which turns out in $R_{F_{\Phi_{assignments}}} = 0{,}925$. Hence, sets S_S and S_F are updated to $S_S = S_S \cup \langle 3{,}0.075\rangle$ and $S_F = S_F \cup \langle 3{,}0.925\rangle$ (see Figure 34). Consequently, the next element in $A_{OMACS_{available}}$ is evaluated, i.e., $\{a_2\}$. As result, the reliability of the optimal assignment, $R_{S_{\Phi_{assignments}}}$, is computed. Because $a_1 \notin \{a_2\}$, $R_{S_{\Phi_{assignments}}} = (1 - asgmtReliability(a_1, r_1, g_3)) * asgmtReliability(a_2, r_2, g_4))$. Hence, $R_{S_{\Phi_{assignments}}} = 0.925 * 0.245$, which gives rise to $R_{S_{\Phi_{assignments}}} = 0.226625$. Because $A_{OMACS_{copy}} \backslash \{a_2\} = a_1$, it implies two things: first a_1 failed to accomplished goal g_3 thru role r_1, therefore removed from $A_{OMACS_{copy}}$, i.e.., $A_{OMACS_{copy}} = \{a_2\}$; and second, goal g_4 is achieved; thus removed from $G_{OMACS_{copy}}$, i.e., $G_{OMACS_{copy}} = \{g_3\}$. Since $\left|A_{OMACS_{copy}}\right| > 1$, i.e., $A_{OMACS_{copy}} = \{a_2\}$, and $\left|G_{OMACS_{copy}}\right| > 1$, i.e., $G_{OMACS_{copy}} = \{g_3\}$, a new transient state of the system is created, i.e., $a_2 \cdot g_3$. Because $a_2 \cdot g_3 \notin S$, sets S and S_T are updated to $S = S \cup \{\langle 1, a_1 \cdot a_2 \cdot a_3 \cdot g_4\rangle, \langle 2, a_1 \cdot a_2 \cdot g_3 \cdot g_4\rangle, \langle 3, a_1 \cdot g_4\rangle, \langle 4, a_2 \cdot g_3\rangle\}$ and$S_T = S_T \cup \{\langle 1, \{0{,}0.001323\}\rangle,\ \langle 2, \{0{,}0.017052\}\rangle,\ \langle\{3, \{2{,}0.056625\}\}\rangle,$
$\langle\{4, \{2, 0.226625\}\}\rangle\}$. Additionally, variable n is updated to 5, i.e., $n = 5$.

```
SerSyst  $reliability := 1.0$;
         for $g_k \in G_{OMACS}$ do
         begin
             $MAX = \infty$;
             for $\langle r_j, g_k' \rangle \in achieves$ do
             begin
                 $RsSerSyst := 1.0$;
                 if $g_k = g_k'$ then
                 begin
                     for $a_i \in A_{OMACS}$ do
                     begin
                         if $\langle a_i, r_j, g_k \rangle \in \phi_{OMACS}$ then
                         begin
                             $RsSerSyst := asgmtReliability(a_i, r_j, g_k)$;
                         end;
                         else
                         begin
                             $RsSerSyst := 0.0$;
                         end;
                     end;
                 end;
                 if $RsSerSyst > MAX$ then
                 begin
                         $MAX := RsSeriesSys$;
                 end;
             end;
             $reliability := reliability * MAX$;
         end;
         $S_S = S_S \cup \langle parent, reliability \rangle$;
         $S_F = S_F \cup \langle parent, 1.0 - reliability \rangle$;
```

Figure 32. Procedure $ParalSyst$ written in Pidgin (see Appendix C).

Accordingly, the procedure $AMC - Spec$ is invoked for fourth time with parameters $A_{OMACS} = \{a_2\}$, $G_{OMACS} = \{g_3\}$, and $parent = (n-1)$, i.e., $parent = 4$. Since $|A_{OMACS}| = 1$ and $|G_{OMACS}| = 1$, sub-procedure $SeriesSystem$ is invoked and two cases are to be evaluated: success and failure. Hence, by Eq. 3.11, $R_{S_{\phi_{assignments}}} = 0.245$. Also, by Eq. 3.13, $R_{F_{\phi_{assignments}}} = 1 - 0.245$, which turns out in $R_{F_{\phi_{assignments}}} = 0{,}755$. Hence, sets S_S and S_F are updated to $S_S = S_S \cup \{4{,}0.245\}$ and $S_F = S_F \cup \{4{,}0.755\}$ (see Figure 34). Afterwards, the last element in $A_{OMACS_{available}}$ is evaluated, i.e., $\emptyset$. As result, the reliability of the optimal assignment $R_{S_{\phi_{assignments}}}$ is computed. Because $a_1, a_2 \notin \emptyset$, $R_{S_{\phi_{assignments}}} = (1 - (1 - asgmtReliability(a_1, r_1, g_3)) * (1 - asgmtReliability(a_2, r_2, g_4)))$. Hence, $R_{S_{\phi_{assignments}}} = 1 - (0.925 * 0.755)$, which gives rise to $R_{S_{\phi_{assignments}}} = 0.301625$.

Furthermore, $R_{F_{\phi_{assignments}}} = 1 - R_{S_{\phi_{assignments}}}$; hence, $R_{F_{\phi_{assignments}}} = 0.698375$. Since $A_{OMACS_{copy}} \backslash \emptyset = \{a_1, a_2\}$, it implies two things: first a_1 and a_2 failed to accomplished goals g_3 and g_4 thru roles r_1 and r_2, respectively, therefore removed from $A_{OMACS_{copy}}$, i.e.., $A_{OMACS_{copy}} = \emptyset$; and second, neither goal g_3 nor goal g_4 is achieved; thus $G_{OMACS_{copy}} = \{g_3, g_4\}$. Since, $A_{OMACS_{copy}}$ is empty and $G_{OMACS_{copy}}$ still contains elements, this implies that our system has reached a failure state, therefore set S_F is updated to $S_F = S_F \cup \{\langle 1{,}0.648092\rangle, \langle 2{,}0.698375\rangle, \langle 3{,}0.925\}\rangle, \langle 4{,}0.755\}\rangle\}$ (see Figure 34). The procedure described before is applied for the rest of the elements in $A_{OMACS_{available}}$, i.e., $\{a_2, a_3\}$, $\{a_1, a_3\}, \{a_1\}, \{a_2\}, \{a_3\}, \emptyset$[26]. As result, Figure 35 - Figure 40 show the branching result of evaluating elements $\{a_2, a_3\}, \{a_1, a_3\}, \{a_1\}, \{a_2\}, \{a_3\}$, and $\emptyset$, respectively.

Finally, after computing sets S_T, S_S, and S_F, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ computes the steady state, $x_n^{(k)}$, of P_{markov} (steps $st7$, $st8$, $st9$, $st10$, $st11$, $st12$, $st13$, and $st14$ and loop $lp1$). First, $\phi_{OMACS}toR_{\phi_{OMACS}}$ adds two new elements to set S_s (see Figure 41). That is, $S_S = S_S \cup \langle 13{,}1.000000\rangle \cup \langle 14{,}0.000000\rangle$ (steps $st7$ and $st8$, Figure 26). Similarly, Algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ adds two new elements to set S_F (see Figure 42). Namely, $S_F = S_F \cup \langle 13{,}0.000000\rangle \cup \langle 14{,}1.000000\rangle$ (steps $st9$ and $st10$, Figure 26). Consequently, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ creates array x, which represents $x_n^{(k)}$, with size $n + 2$, i.e., 14. Afterward variable δ, the result after the $k + 1$ iteration, and ε, the accuracy of the result after $k + 1$, are initialized. That is, $\delta = 0$ and $\varepsilon = 0.000001$. Subsequently, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ evaluates whether or not $\delta > \varepsilon$. If true, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ computes the product, i, of array x and P_{markov} (see Figure 33). It can be noted that P_{markov} is implicitly constructed in terms of sets S_T, S_S, and S_F (Figure 43). Table 9 shows the resulting steady state of P_{markov} after 3 iterations, i.e., $x_n^{(3)}$. Notice that the index of interest in x is 13, i.e., success state s_{13}. This state represents the reliability of the organization-based multiagent systems in terms of the assignment set

[26] First invocation of the procedure $NonSerParalSyst$.

ϕ_{OMACS} under evaluation (step $st15$, Figure 26). Thus, the reliability of the system is less than 1%, i.e., 0.6%. Additionally, the unreliability of the system can be defined as follows

$$R_{F_{\phi_{OMACS}}} = 1 - R_{S_{\phi_{OMACS}}} \tag{3.13}$$

Hence, with a probability of 99.4%, the current system configuration is prone to failure.

input: S_S, S_F, S_T, x
comment: Set S_S stores elements of the form $\langle i, p_{i\ success} \rangle$, where $p_{i\ success}$ represents the transition probability from state i to state $success$ set S_F stores elements of the form $\langle i, p_{i\ failure} \rangle$, where $p_{i\ failure}$ represents the transition probability from state i to state $failure$; and; S_T stores elements of the form $\langle j, \langle i, p_{ij} \rangle\rangle$, where p_{ij} represents the transition probability between transient states i to j; and x the probability vector of P_{Markov}, i. e., $x_n^{(k)}$
output: x_{new}, the probability vector of P_{Markov}, i. e., $x_n^{(k+1)}$

```
Procedure matrixProduct($S_T, S_S, S_F, x$):
begin
st1:  $x_{new}[1: n+2]$;
st2:  $rpi := \emptyset$;
lp1:  for $i = 1$ to $n + 2$ do
      begin
          $x_{new}[i] := 0$;
      end;
lp2   for $i = 1$ to $n$ do
      begin
          if $i = 1$ then
          begin
              $x_{new}[i] := 0$;
              $rpi := rpi \cup \langle i, x[i] \rangle$;
          end;
          else
          begin
              if $\langle i, \gamma \rangle \notin S_T$ then
              begin $x_{new}[i] := 0$; end;
              else
              begin
                  if $x[i]! = 0$ then
                  begin $rpi := rpi \cup \langle i, x[i] \rangle$; end;
                  else
                  begin
                      for $\langle j, value \rangle \in \gamma$ do
                      begin $x_{new}[i] := x_{new}[i] + (x_{new}[j] * value)$; end;
                  end;
              end;
          end;
      end;
st3:  $rpi := rpi \cup \langle n, x[n] \rangle$;
st4:  $rpi := rpi \cup \langle n+1, x[n+1] \rangle$;
st5:  $k := top$;
st6:  $sum := 0.0$;
lp3:  for $\langle i, value_{rpi} \rangle \in rpi$ do
      begin
          if $\langle i, value_{S_S} \rangle \in S_S$ then
          begin $sum := sum + x[i] * value_{S_S}$; end;
      end;
st7:  $x_{new}[k] := sum$;
st8:  $k := k + 1$;
lp4:  for $\langle i, value_{rpi} \rangle \in rpi$ do
      begin
          if $\langle i, value_{S_F} \rangle \in S_F$ then
          begin $sum := sum + x[i] * value_{S_F}$; end;
      end;
st9:  $x_{new}[k] := sum$;
st10:  $rpi = \emptyset$;
end;
```

Figure 33. Procedure $matrixProduct$ written in Pidgin (see Appendix C).

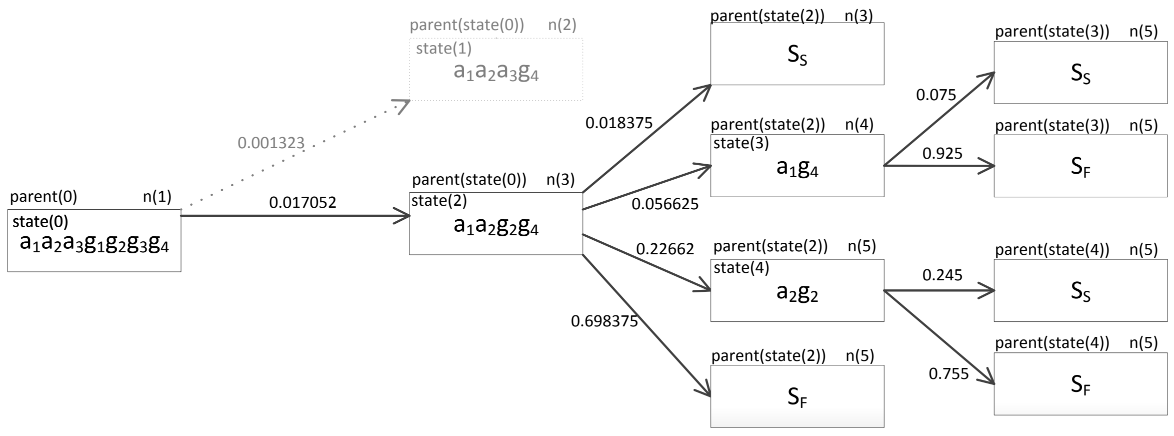


Figure 34. Branching of State 2, i.e., $a_1 \cdot a_2 \cdot g_2 \cdot g_4$

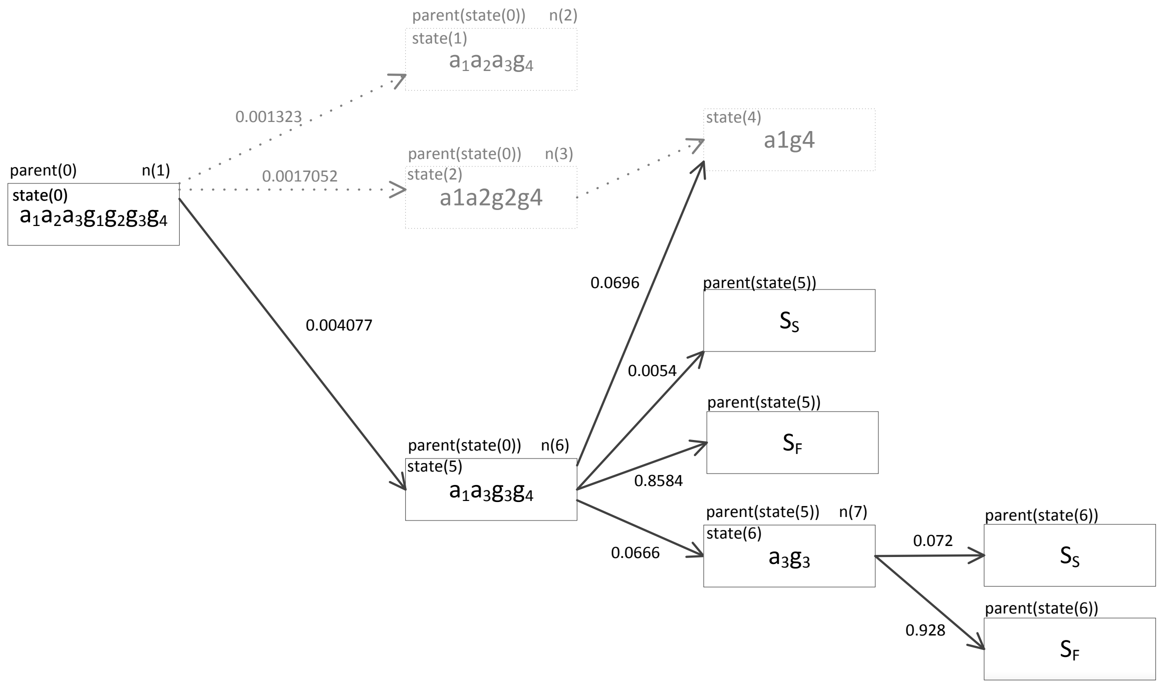


Figure 35. Branching of State 3, i.e., $a_1 \cdot a_3 \cdot g_3 \cdot g_4$

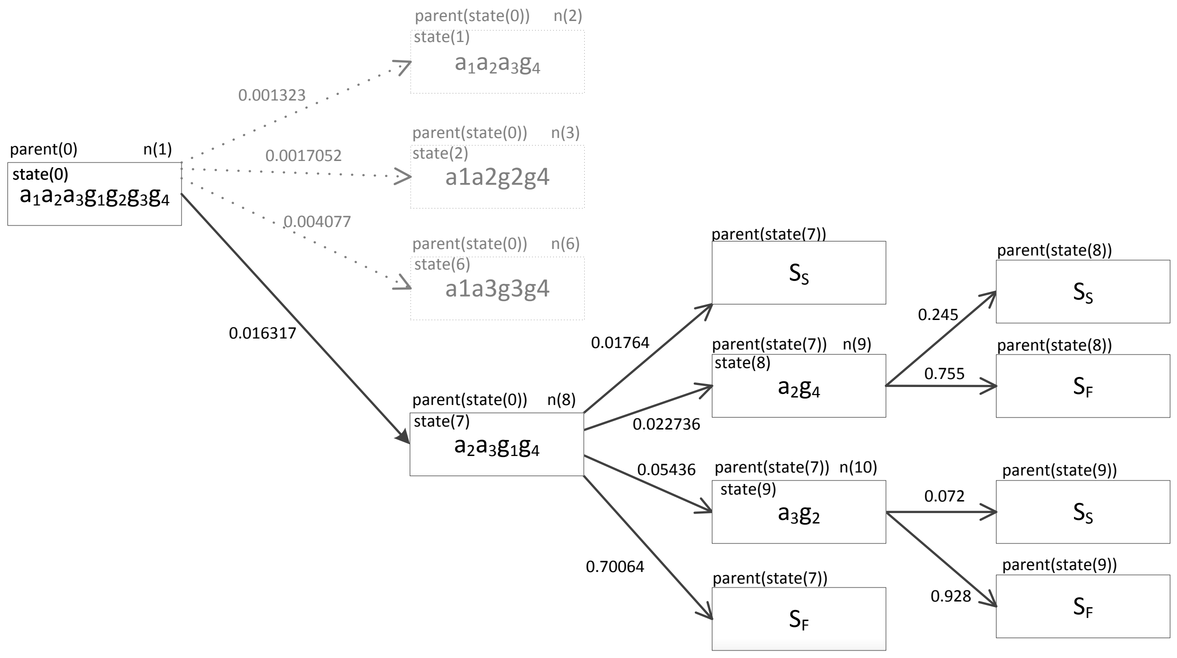


Figure 36. Branching of State 7, i.e., $a_2 \cdot a_3 \cdot g_1 \cdot g_4$

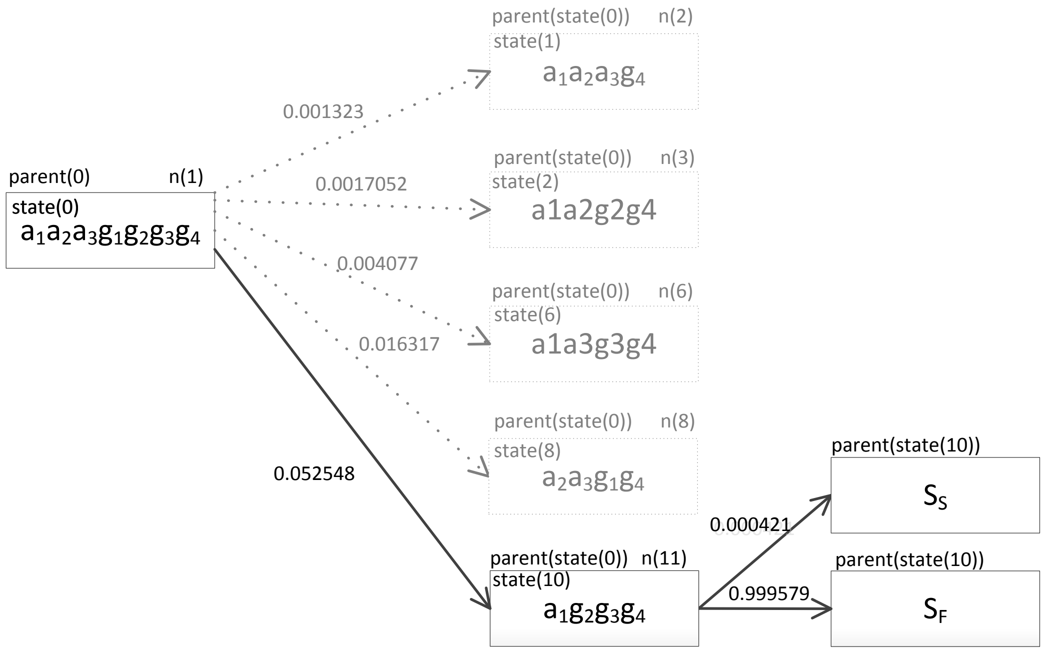


Figure 37. Branching of State 10, i.e., $a_1 \cdot a_2 \cdot g_3 \cdot g_4$

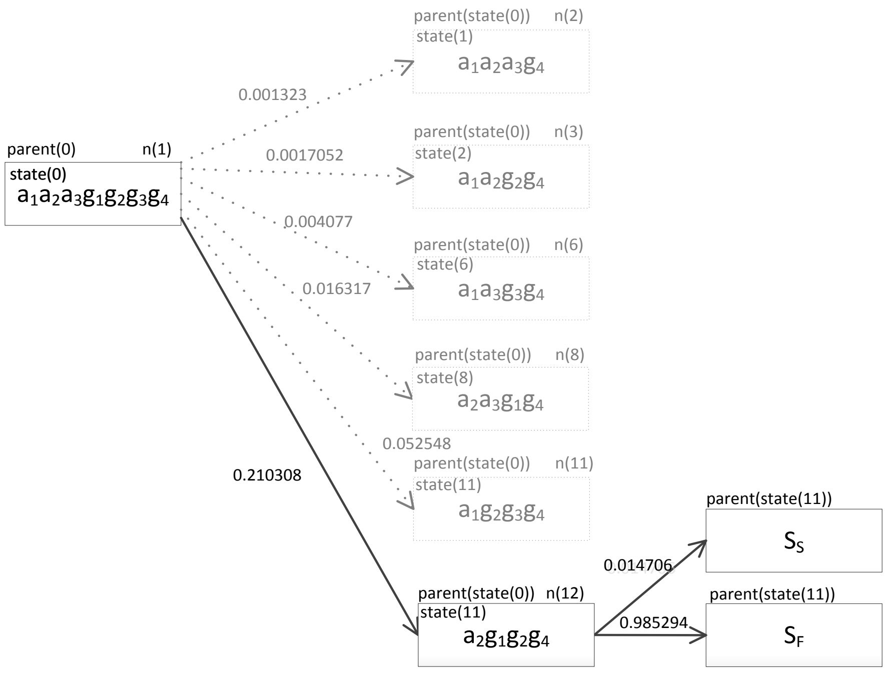


Figure 38. Branching of State 11, i.e., $a_2 \cdot g_1 \cdot g_2 \cdot g_4$

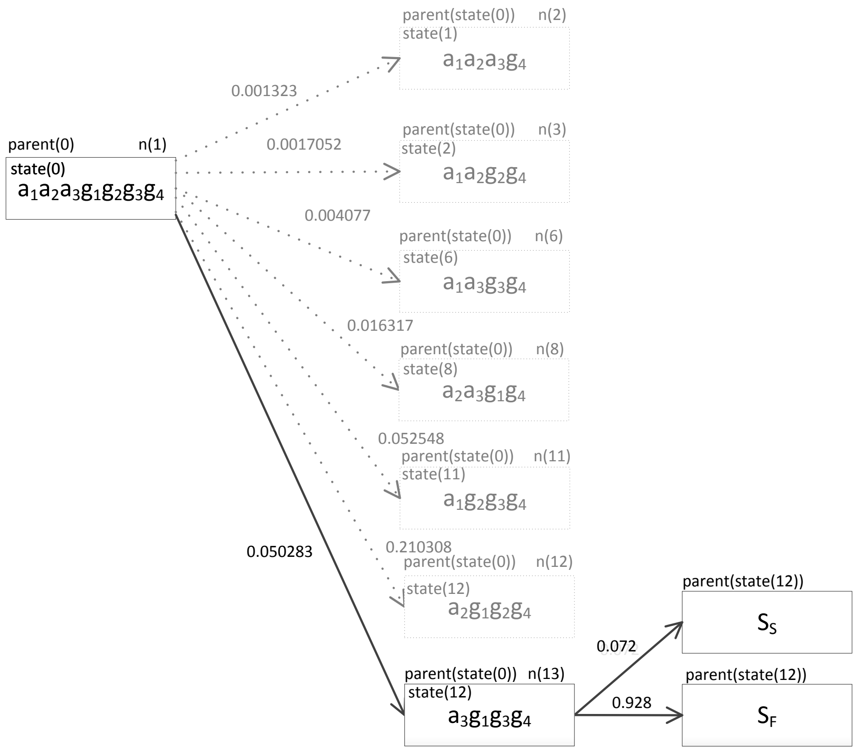


Figure 39. Branching of State 12, i.e., $a_3 \cdot g_1 \cdot g_3 \cdot g_4$

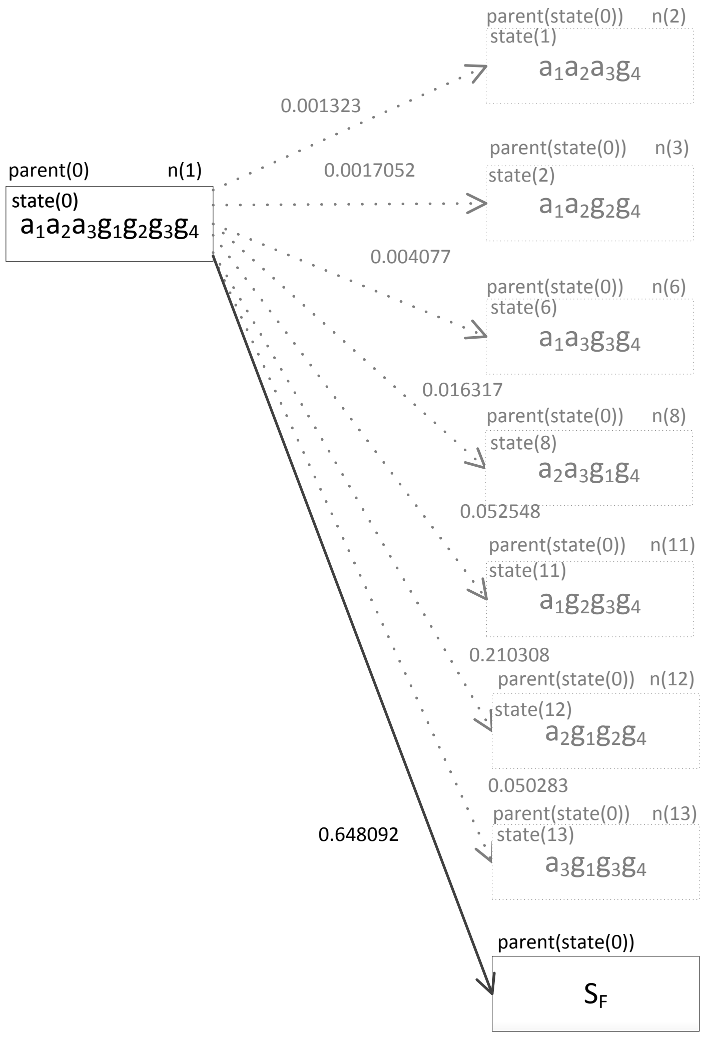


Figure 40. Branching of one-step transition from initial State *0*, i.e. $a_1 \cdot a_2 \cdot a_3 \cdot a_4 \cdot g_1 \cdot g_2 \cdot g_3 \cdot g_4$, to State *Failure*

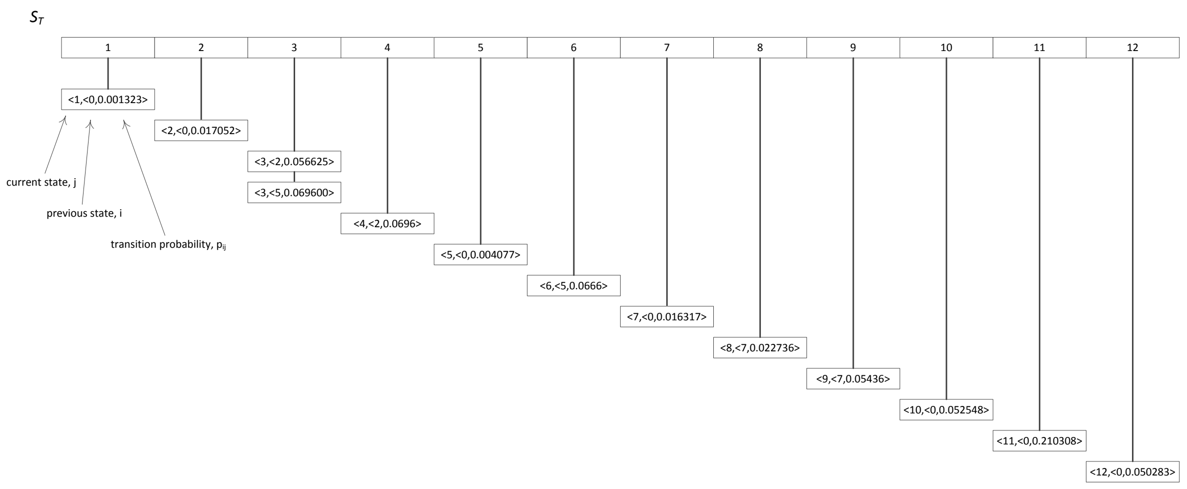


Figure 41. Structure of set S_T seen as an adjacent list

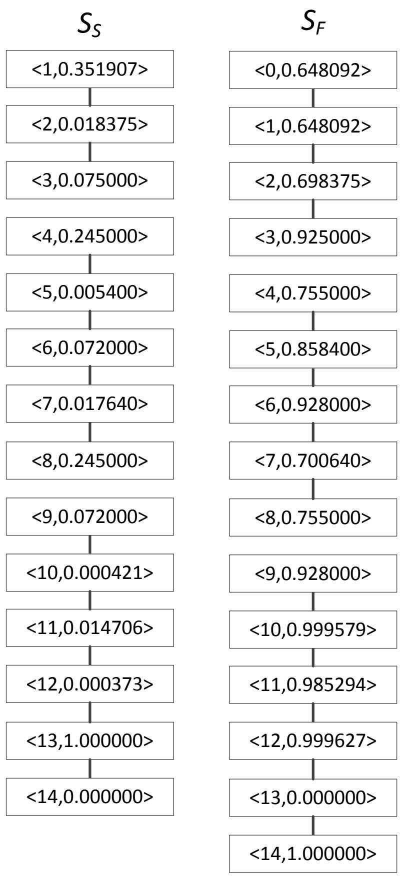


Figure 42. Structure of sets S_S and S_T seen as an adjacent list

$$\boldsymbol{P_{markov}} = \left[\begin{matrix}
0.000000 & 0.001323 & 0.017052 & 0.000000 & 0.000000 & 0.004077 & 0.000000 & 0.016317 & 0.000000 & 0.000000 & 0.052548 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.056625 & 0.226625 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.069600 & 0.000000 & 0.000000 & 0.066600 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.227360 & 0.054360 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0
\end{matrix}\right.$$

Figure 43. Transition Matrix $\boldsymbol{P_{markov}}$.

Table 9. Probability distribution, after 3 iterations, given the initial probability vector, $x_n^{(0)}$, [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

k	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13} *success*	s_{14} *failure*
1	0,000	0,001	0,017	0,000	0,000	0,004	0,000	0,016	0,000	0,000	0,053	0,210	0,050	0,000	0,648
2	0,000	0,000	0,000	0,001	0,004	0,000	0,000	0,000	0,004	0,001	0,000	0,000	0,000	0,004	0,986
3	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,006	0,994

3.6 Assessment of Organization based Multi-agent System Design by the Absorbing Markov Chain Model Method

To empirically evaluate the reliability of the different agent-based organization designs computed by algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$, we have developed a simulation that steps through the design of a CRST application. To measure the system reliability, a Bernoulli process is followed [7]. For each assignment set, ϕ_{OMACS} (see Table 8), a randomly system goal, i.e., $g_k \in G_{OMACS}$, is selected. Subsequently, the reliability of the best available assignment, i.e., $asgmtReliability(a_i, r_{j,} g_k)$, is calculated. The best assignment defines what is the reliability of an agent, $a_i \in A_{OMACS}$, for achieving a goal, $g_k \in G_{OMACS}$, playing a role, $r_k \in R_{OMACS}$ (see Eq. 3.10). Afterwards, $asgmtReliability(a_i, r_{j,} g_k)$ is compared to a random variable X, which is uniformly distributed (Java 1.7.0_55 pseudo-random number generator). If X is greater than $asgmtReliability(a_i, r_{j,} g_k)$, agent a_i is removed from A_{OMACS}, it is assumed that a_i failed to achieve goal g_k; otherwise, g_k is removed from G_{OMACS}. This process will continue until either A_{OMACS} or G_{OMACS} is empty. If G_{OMACS} is empty, 0 is returned; otherwise, 1 is returned. Noticed that 0 represents a success and 1 represents a failure for each trial. 5000 trials are tested for the Bernoulli process. Finally, the resulting probability of success is calculated as average of the trials in order to level out variations caused by the random generator used to simulate success or failure of an assignment.

Figure 44 reveals that the reliability of System #1 (refer to Figure 16) ranges between a max of 0.626% and a min of 0.055% with a media of 0.41386% and a standard deviation of 0,2368%. As result, system # 1, in overall, can be considered faulty (see in Figure 16 the behavior of the first hundred configurations).

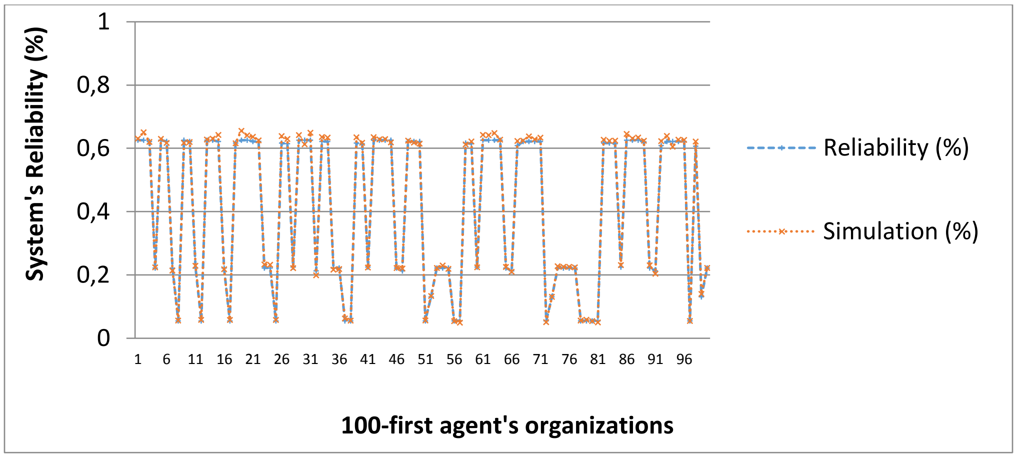


Figure 44. Results for System 1

Additionally, two other hypothetical CRST systems are specified: System #2 and system #3 (see Figure 45 and Figure 47). Figure 46 shows that the reliability of System #2 ranges between a max of 2.993% and a min of 1.409% with a media of 2.3889% and a standard deviation of 0,4681%. As consequence, System #2, in overall, can also be considered faulty.

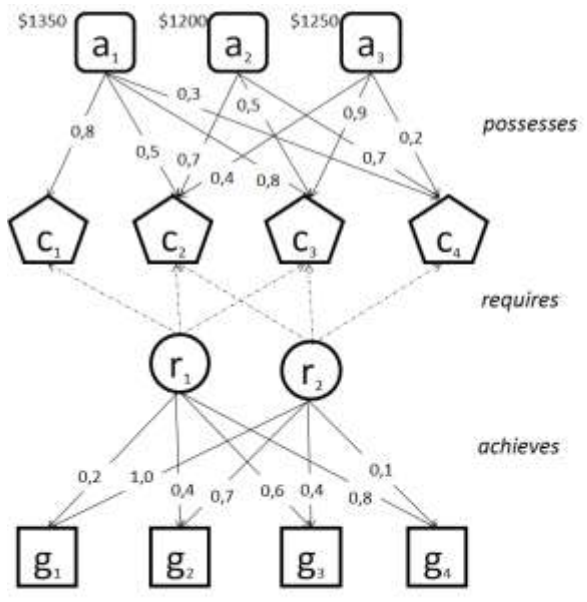


Figure 45. Overview of the CRST Organization # 2.

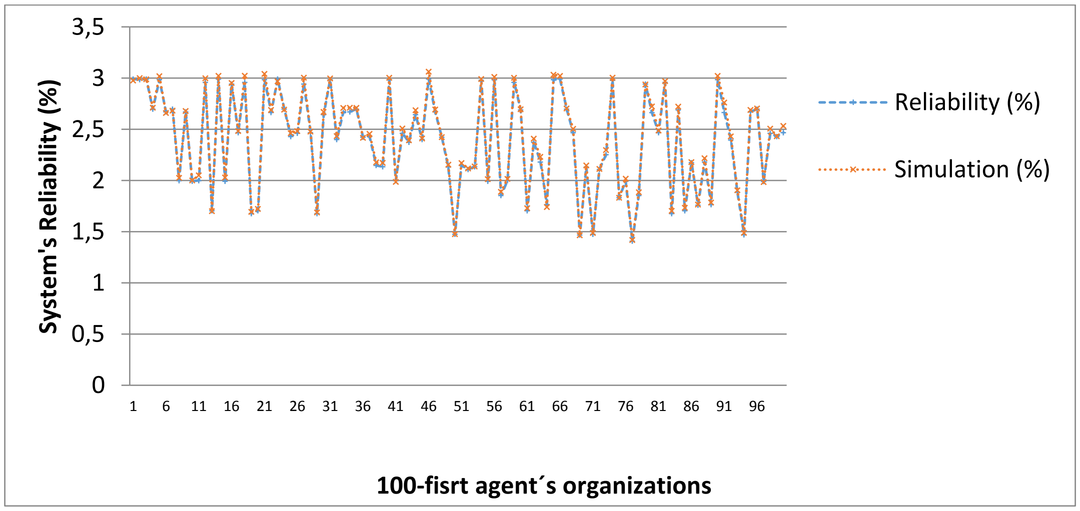


Figure 46. Results for System 2

Finally, Figure 48 illustrates the reliability of System #3, which ranges between a max of 97.347% and a min of 77.536% with a media of 92.5631% and a standard deviation of 6.7159%. Respectively, System #3, in overall, can be considered reliable. It can be noted that the reliability calculated of most of the subset of all the potential assignments,$\phi_{OMACS_{System\ \#\ 3}}$, is greater that 90%. These outcomes give rise to the formulation of new research questions. For example, why does the reliability of some assignment sets fall below a prescribed target value? This observation can be seen in Figure 48 where roughly the reliability of 16/100 of the assignment set falls below 80%.

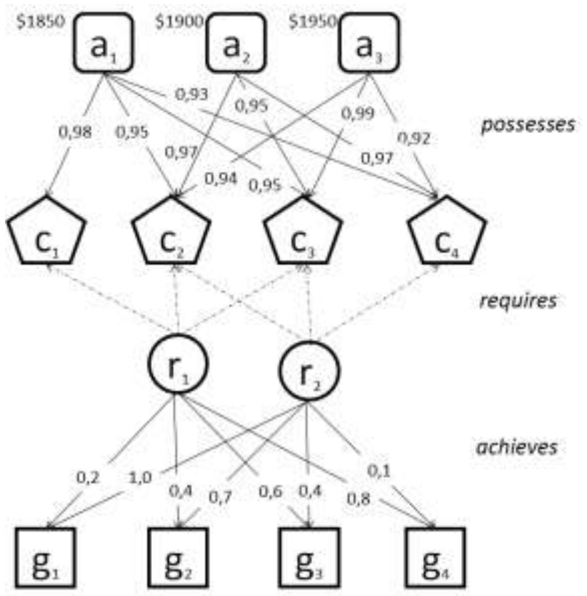


Figure 47. Overview of the CRST Organization # 3.

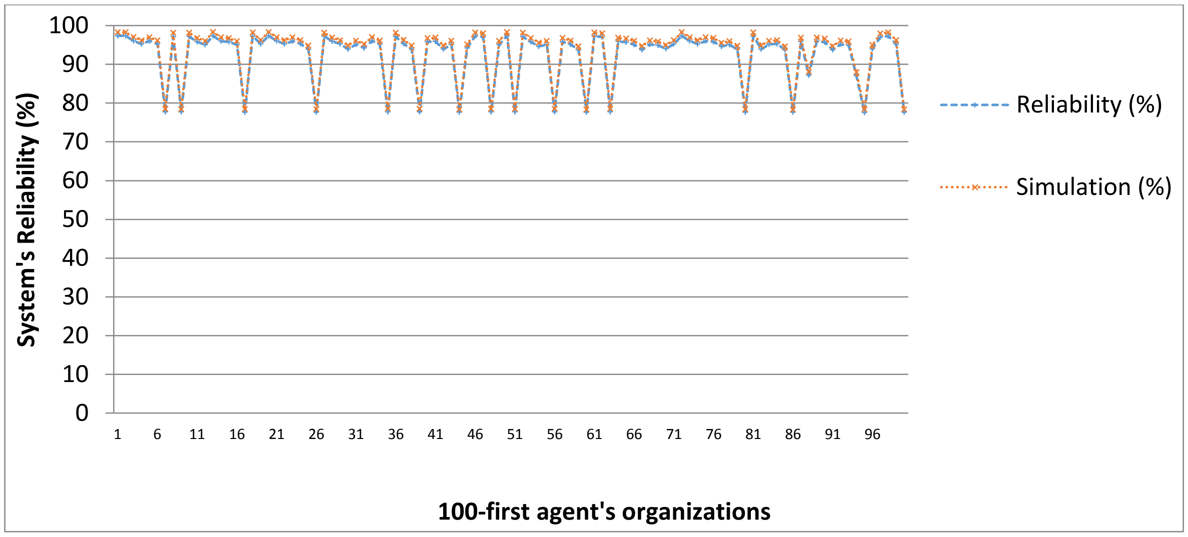


Figure 48. Results for System #3.

Chapter 4. Conclusions and Recommendations for Future Work

In this dissertation, two problems, building-evacuation route planning and organization-based multiagent systems, have been analyzed and modeled by resorting to the P-graph framework. What follows are the significant conclusions drawn as well as recommendations for future work.

4.1 Building-Evacuation-Route Planning

An algorithmic method has been proposed for generating optimal building-evacuation routes and n-best sub-optimal solutions for a building-evacuation problem. The method has been crafted by transforming a building-evacuation problem into a PNS_T problem and solving it via the algorithms and software of the P-graph framework. The efficacy of the proposed method has been illustrated with several examples in which the optimal and sub-optimal evacuation routes emerge in ranked order by defining the objective function as the cost of reaching a safety point in time t, where $0 < t \leq T$. The results obtained indicate that the proposed method outperforms current optimization models [41,42].

Nevertheless, the computational performance of the proposed method should be compared to those of other generic commercial optimization tools, i.e., Cplex (http://goo.gl/375IV5) and Gams (http://goo.gl/kfZAyM), in terms of computational time and storage. This comparison should also include the appraisal of parallelized versions of the algorithm ABB as introduced in [101,111]. Moreover, a proof-of-concept system should be developed to assess the suitability of the proposed method in real-life scenarios where individuals are in constant movement inside the building, the building conditions are required to be captured periodically, and the behavior of the individual might be crucial in following directions [12,49,92]. In addition, other mathematical models, e.g., linear ordering, should be studied [2,12] to explore the possibility of enhancing the proposed method with some of the features characterizing such mathematical models. These features include individual travel and exposure time as well as time-based risk and evacuation exposure [14,49,50].

4.2 Modeling Organization-based Multiagent Systems Design

The assessment of the n-best organizational-based multiagent system design based on the OMACS framework has been performed by deploying an algorithmic method. The method has been implemented by transforming an organizational-based multiagent system design into a PNS problem and solving it by means of the algorithms and software of the P-graph framework.

The algorithmic method has been illustrated with an instance in which the optimal and sub-optimal organizational-based multiagent system designs emerge in ranked order by defining the objective function as the cost of each design in terms of the oaf function, φ, (see Eq. 3). An optimal solution, however, does not always capture the expected behavior of the organizational-based multiagent system design. To amend this, a second method has been deployed, which is based on absorbing Markov chains and concepts pertaining to the field of systems reliability. The results obtained from both methods have been assessed by simulation.

Finally, we propose the construction of a computational tool for transforming OMACS organizational-based multiagent systems into PNS problems. Our efforts in this regard will be the subject of future contributions.

Chapter 5. Summary of Accomplishments

5.1 Original Contributions

5.1.1Theses

Based on the novel results and scientific contributions presented and illustrated by several case studies (either from the literature or hypothetical) in the previous chapters, the following theses represents the basic discoveries of the present dissertation.

With regards to the building-evacuation –route planning problem, a method and software are proposed:

1. The building-evacuation routes are represented by a P-graph, which gives rise to a time-expanded process-network-synthesis ($\mathrm{PNS}_{\mathrm{T}}$) problem.

2. A PNS_T problem takes into account the temporal dimension inherent to the building evacuation problem in terms of the evacuation time, specifically its upper bound T.

3. A PNS_T problem can be algorithmically solved according to the P-graph framework, where each location and passage in the building are given by a set of attributed to be taken into the evacuation-route-planning.

4. The evacuation time (which also let us computes the evacuation routes and scheduling of evacuees on each route) is calculated as a minimum cost of the corresponding PNS_T.

5. In addition to the globally optimal solution the P-graph framework provides the n-best sub-optimal solution.

6. The validity of the proposed method and software is illustrated by several examples taken from the literature.

With respect to the modeling organization-based multiagent systems problem:

7. At the outset, the design of organization-based multiagent systems is proposed according to the framework of Organization Model for Adaptive Complex Systems (OMACS).

8. This design model is transformed into a process-network model, i.e., P-graph. The resultant process-network model in conjunction with the P-graph-based methodology give rise to:

    a. the maximal structure of the process network, comprising all the combinatorially feasible structures, i.e., OMACS-based design configurations, capable of yielding the specified products from the specified raw material;
    b. every combinatorially feasible structure of the process of interest; and,
    c. the optimal structure of the network, i.e., the optimal OMACS-based design configuration.

9. 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. However, the outcome of thesis 9 renders it possible to rule out thesis 8.c. That, it is not always the rule that the higher the cost of the OMACS-based design configuration, the better the performance of the agent-based organization.

To overcome this result, in this dissertation the algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ is introduced.

10. The aim of algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ is to transform an organization-based multiagent system assignment set, ϕ_{OMACS}, into an absorbing-markov chain, P_{markov}, and compute its steady state, $x_n^{(k)}$. That is to say, algorithm $\phi_{OMACS}toR_{\phi_{OMACS}}$ is to evaluate whether or not ϕ_{OMACS} leads the organization-based multiagent system into one of the absorbing states, i.e., either the agents' organization achieves all its goals (success state) or fails to (failure state).

11. The validity of the proposed method and software is illustrated by examining a hypothetical example extracted from the literature.

5.2 List of Publications

The publications in book, book chapters, international journals and peer reviewed international conference papers which are related to this dissertation are listed below.

Book

1. García-Ojeda, Juan C. "GADMAS: Combinando el Modelado Organizacional con Meta-modelos Gráficos en el Desarrollo de Sistemas Multiagente," Editorial Académica Española. pp. 140, 2012 (in spanish).

Book Chapters

1. DeLoach, Scott A., and García-Ojeda, Juan C. "The O-MaSE Methodology," In: Massimo Cossentino, Vincent Hilaire, Ambra Molesini, Valeria Seidita (Eds.): *Handbook on Agent-Oriented Design Processes.* Springer-Verlag: Berlin, 2014, pp. 253 – 286.

Refereed International Journals

1. Garcia-Ojeda, J. C., Bertok, B., Friedler, F., Argoti, A., Fan, L.T. “A Preliminary Study of the Application of the P-graph Methodology for Organization-based Multiagent System Designs: Assessment” *Acta Polytechnica Hungarica* 12, (2) (2015), 103-122. (IF=0.471).
2. Garcia-Ojeda, J. C., B. Bertok, F. Friedler., L.T. Fan. “Building-Evacuation-Route Planning via Time-Expanded Process-Network Synthesis,” *Fire Safety Journal*, 61 (2013) 338 – 347. (Impact Factor=0.957).
3. Garcia-Ojeda, J. C., B. Bertok, F. Friedler. “Planning evacuation routes with the P-graph framework,” *Chemical Engineering Transactions*, 29, (2012), 1531-1536. (IF=1.03).
4. García-Ojeda, Juan C. “On Building Evacuation Route Planning by Resorting to P-graph,” *Revista Colombiana de Computación*, 12, (1) (2011), 111-125.

Refereed Proceedings Articles

1. García-Ojeda, Juan C., Bertok, Botond, Friedler, Ferenc, Argoti, Andres. “Identifying Evacuation Routes via the P-grpah Methodology,” In: *Proceedings of the 10th Colombian Congress on Computation* (Bogotá, Colombia). 10CCC 2015.
2. García-Ojeda, Juan C., Bertok, Botond, Friedler, Ferenc, Fan, L.T. “On Combining the P-graph Framework and Absorbing Markov Chains for Assessing the Reliability and Cost of Organization-based Multiagent System Design Models,” In: *Proceedings of the 6th Veszprem Optimisation Conference: Advanced algorithms* (Veszprem, Hungary). VOCAL 2014.
3. García-Ojeda, Juan C., Bertok, Botond, and Friedler, Ferenc. “Multi-criteria Analysis of Building Evacuation Route Planning by Resorting to the P-graph Framework,” In: *Proceedings of the 5th Veszprem Optimisation Conference: Advanced algorithms* (Veszprem, Hungary). VOCAL 2012.
4. García-Ojeda, Juan C., Bertok, Botond, and Friedler, Ferenc. “Planning evacuation routes with the P-graph framework,” In: *Proceedings of the 15th International Conferences on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction* (Prague, Czech Republic). PRES 2012.

5.3 List of Publications in other Research Topics

The publications in book chapters, international journals and peer reviewed international conference papers which are related to other research topics distinct to the addressed in this dissertation are listed below.

Refereed International Journals

1. Pimentel Losada, J.P.A., Heckl, I., Bertok, B., Friedler, F., García-Ojeda, J.C., Argoti, A. "Process-Network Synthesis for Benzaldehyde Production, P-Graph Approach"
   Chemical Engineering Transactions, 45, (2015), 1369-1374. (IF=1.03).

2. García-Ojeda, Juan C. "Measurement of Tailored Agent-oriented Design Processes by Resorting to Flow Graphs: A Preliminary Investigation," *Revista Colombiana de Computación*, 11, (2) (2010), 94 -115.

3. García-Ojeda, Juan C., and DeLoach, Scott A. "The O-MaSE Process: a Standard View," *CEUR Workshop Proceedings*. 627 (2010), II—DPDF 55 -66.

4. DeLoach, Scott A., and García-Ojeda, Juan C. "O-MaSE: An Customizable Approach to Designing and Building Complex, Adaptive Multiagent Systems," *International Journal on Agent-Oriented Software Engineering*. 4 (3) (2010), 244 – 280.

5. Arenas, Álvaro E., García-Ojeda, Juan C., and Pérez-Alcázar, Jose de J. "On combining organisational modelling and graphical languages for the development of multiagent systems," *Integr. Comput.-Aided Eng.* 11, 2 (2004), 151-163. (IF=4.698, 2015)

Book Chapters

1. Castellanos, H.C., García-Ojeda, J.C., Calier, F.R.: "Propuesta de arquitectura para la interoperabilidad de la historia clínica electrónica en Colombia," In: Gustavo Velásquez Quintana (Ed.): *Tecnología e innovación: Aplicaciones para el desarrollo de la ciencia y la sociedad*. Universidad Nacional Abierta y a Distancia: Bogotá, DC, 2014, pp. 57 – 67.

2. Arenas, Daniel, Sandoval, Edward, García-Ojeda, Juan C., Gómez, Martha, y Cáceres, Claudia. "Servicios de Localización, Georeferenciación, y Mensajería a través de la Computación Móvil." In: Aguilar Vera , Raúl A., Díaz Mendoza, Julio C., Gómez Cruz, Gerson E., y Bohórquez, Edwin León (Eds.): Ingeniería de Software e Ingeniería del Conocimiento: Tendencias de Investigación e Innovación Tecnológica en Iberoamérica. AlfaOmega Grupo Editor, 2010, pp. 310-320.

3. García-Ojeda, Juan C., DeLoach, Scott A., Robby, Oyenan, Walamitien H., and Valenzuela, Jorge. "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.

4. García-Ojeda, Juan C., Arenas, Álvaro E., and Pérez-Alcázar, José de J. "Paving the Way for Implementing Multiagent Systems: Refining Gaia with AUML," In: Jörg P. Müller, Franco Zambonelli (Eds.): *Agent-Oriented Software Engineering VI, 6th International Workshop, AOSE 2005, Utrecht, The Netherlands, July 25, 2005. Revised and Invited Papers*, LNCS 3950, 179-189, Springer-Verlag: Berlin, 2006.

5. Barrera-Sanabria, G., Arenas-Seleey, D., García-Ojeda, Juan C., Méndez-Ortiz, F. Designing Adaptive Educational Web Sites: General Framework. In: Kinshuk, Chee-Kit Looi, Erkki Sutinen, Demetrios G. Sampson, Ignacio Aedo, Lorna Uden, Esko Kähkönen (Eds.): *Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICALT 2004, 30 August - 1 September 2004, Joensuu, Finland*, 973-977, IEEE Computer Society, 2004.

6. Barrera-Sanabria, Gareth, Arenas-Seleey, Daniel, García-Ojeda, Juan C., and Méndez-Ortíz, Freddy. "Modelling Intelligent Agents for Adaptive Educational Web Sites," In: Knowledge – based Software Engineering: Proceedings of the Sixth Joint Conference on Knowledge-based Software Engineering. Stefanuk, V. and Kaijiri, K. (Eds.) IOS Press, 2004.

7. Pérez-Alcázar, José de J., Arenas, Álvaro E., Barrera-Sanabria, G., and García-Ojeda, Juan C. Hacia una Ingeniería de Software Orientada a Agentes. En: Cecilia Maria Lasserre: Oportunidades de Cooperación en Ingeniería del Software e Ingeniería del Conocimiento: Investigación en Iberoamérica. 3as JIISIC, 39 – 44, EdiUnju, 2003.

Refereed Proceedings Articles

1. García-Ojeda, Juan C., " Aplicación de las TIC en Soluciones para la Captura Digital de Clases Presenciales y Virtuales en Instituciones de Educación Superior". In: *Proceedings of World Engineering Education Forum* (Cartagena, Colombia). 2013.

2. Castellanos Granados, Hernán Camilo, García-Ojeda, Juan C., Rueda Calier, Fabio. "Propuesta de arquitectura para la interoperabilidad de la historia clínica electrónica en Colombia". In: *Memorias del Encuentro Nacional de ingenierías* (Cali, Colombia). 2013.

3. Garcia Angarita, Maritza Andrea, García-Ojeda, Juan C, and Alvarez Lopez, Ramón Antonio. "Moodle Como Apoyo Académico en la Educación Técnica,"

In: *Memorias del 4to congreso Internacional de Ambientes Virtuales de Aprendizaje Adaptativos y Accesibles* (Cartagena, Colombia). CAVA 2012.

4. Tobar, David, Mendez, Anderson H., and García-Ojeda, Juan C. "ATenEa – Aplicación de las Tecnologías de la Información y de la Comunicación en Soluciones para la Captura Digital de Clases Presenciales y Virtuales en Instituciones de Educación Superior como Herramienta Tecnológica para la Generación de Impacto en la Enseñanza," In: *Memorias del 4to congreso Internacional de Ambientes Virtuales de Aprendizaje Adaptativos y Accesibles* (Cartagena, Colombia). CAVA 2012.

5. García-Ojeda, Juan C., and Briceño, Wilson. "Ant's Business Game: Una propuesta de Simulador Gerencial en el Contexto de la Pequeña y Mediana Empresa en Colombia," In: *Proceedings of the International Conferences in Economics, Management, and Accounting* (Bucaramanga, Colombia). ICEMA 2012.

6. Garcia Prada, Andrea Patricia, and García-Ojeda, Juan C. "Vista Grafica para la Administración de Procesos Personalizados basados en Agentes para la Herramienta *agentTool Process Editor (APE)*: Soportando la Técnica *Earned Value Analisis*," In: *Memorias Sexto Congreso Colombiano de Computación* (Manizales, Colombia). 6CCC, 2011.

7. García-Ojeda, Juan C., and DeLoach, S. A. "The O-MaSe Process: A Standard View," In: *Proceedings of the IEEE FIPA Workshop on Design Process Documentation and Fragmentation* (Lyon, France). MALLOW 2010, 2010.

8. García-Ojeda, Juan C., DeLoach, S. A., and Robby 2009. "agentTool III: From Process Definition to Code Generation," In: *Proceedings of the 8th international Conference on Autonomous Agents and Multiagent Systems - Volume 2* (Budapest, Hungary, May 10 - 15, 2009), 1393-1394, 2009.

9. García-Ojeda, Juan C., DeLoach, S. A., and Robby 2009. "agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes," In: *Proceedings of the 2009 ACM Symposium on Applied Computing* (Honolulu, Hawaii). SAC '09, 707-714, 2009.

10. García-Ojeda, Juan C., Pérez-Álcazar, José de J., and Arenas, A. E. "Extending the Gaia Methodology with Agent-UML," In: *Proceedings of the Third international Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3* (New York, New York, July 19 - 23, 2004), 1456-1457, 2004.

11. Arenas Seleey, Daniel, Barrera Sanabria, Gareth, Méndez Ortíz, Freddy, García-Ojeda, Juan C. "Designinig Adaptative Educational Web Sites General framework," In: *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies IEEE Computer Society*. Finland, 2004.

12. Pérez-Alcázar, José de J., Arenas, Álvaro E., Barrera-Sanabria, G., and García-Ojeda, Juan C. "Hacia una Ingeniería de Software Orientada a Agentes," En: sesión de cooperacion cientifica, 3a Jornada Iberoamericana de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'03), Valdivia, Chile, November 26-28, 2003.

13. García-Ojeda, Juan C., Pérez-Álcazar, José de J., and Arenas, A. E. "Applying Gaia and AUML to the Selective Dissemination of Information on the Web," In: *Proceedings of the 4th Iberoamerican Workshop on Multiagent Systems, Agent Technology and Software Engineering at Iberamia 2002.*

14. García-Ojeda, Juan C., Pérez Alcázar, José de J., y Arenas, Álvaro E. "Aplicación de una Metodología de Desarrollo de Sistemas Multiagente en la Diseminación Selectiva de Información en la Web," En: Memorias del II Congreso Iberoamericano de Telemática (CITA'02). September, 2002. (In Spanish)

Non-refereed Proceedings Articles (In Spanish)

1. García-Ojeda, Juan C., Briceño, Wilson, y Mendoza, Javier. "Construcción de un Modelo de Simulación de Gestión para el Desarrollo de Habilidades Gerenciales," En: II Encuentro Departamental de Semilleros de Investigación. Bucaramanga, Colombia, 2005.

2. García-Ojeda, Juan C., Briceño, Wilson, y Mendoza, Javier. "Construcción de un Modelo de Simulación de Gestión para el Desarrollo de Habilidades Gerenciales," En: VII Encuentro Nacional de Semilleros de Investigación. Cartagena, Colombia, 2004.

3. Arenas-Seleey, Daniel, Barrera-Sanabria, Gareth, Díaz, Ricardo, García-Ojeda, Juan C., Méndez-Ortiz, Freddy y Sarmiento, Román E. "Hacia una Web Adaptativa para Portales Web Educativos," En: Sexto Encuentro Nacional de Semilleros de Investigación, en CD-ROM, USACA - Cali, Colombia, 2003.

4. Arenas, Álvaro E., García-Ojeda, Juan C., y Pérez Alcázar, José de J. "Hacia una Ingeniería del Software Orientada a Agentes: Evaluando Gaia y AUML en el Análisis y Diseño de Sistemas Multiagente," En: Sexto Encuentro Nacional de Semilleros de Investigación, en CD-ROM, USACA - Cali, Colombia, 2003.

5. Arenas-Seleey, Daniel, Barrera-Sanabria, Gareth, García-Ojeda, Juan C., Méndez-Ortiz, Freddy y Sarmiento, Román E. "Propuesta de un Framework General para el Diseño de Portales Educativos Adaptativos". En: V foro de investigaciones RIBIECOL, en CD-ROM, UNAB – Bucaramanga, Colombia, 2003.

6. Arenas, Álvaro E., Pérez Alcázar, José de J., Barrera-Sanabria, Gareth, García-Ojeda, Juan C., y Calderón-Benavides, Maritza L. "Experiencias en: Hacia una Ingeniería del Software Orientada a Agentes," En: V Encuentro Nacional de Semilleros de Investigación, en CD-ROM, UPTC - Tunja, Colombia, 2002.

7. García-Ojeda, Juan C., Pérez Alcázar, José de J., y Arenas, Álvaro E. "Aplicación de una Metodología de Desarrollo de Sistemas Multiagente en la Diseminación Selectiva de Información en la Web," En: Tercera Semana Nacional de Ingeniería de Telecomunicaciones, en CD-ROM, USTA - Medellín, Colombia, 2002.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddler River, NJ (1993)

2. Arulselvan, A.: Network Model For Disaster Management. Ph.D. Dissertation, University of Florida (2009)

3. Benjaafar, S., Dooley, K., Setyawan, W.: Cellular Automata for Traffic Flow Modeling. University of Minnesota – Report. Minneapolis, MN (1997).

4. Berlin, G.N.: A Network Analysis of Building Egress System. ORSA/TIMS meeting, p. 8. Washington, DC (1985)

5. Blue, V. J., Adler, J.L.: Using Cellular Automata Micro-simulation to Model Pedestrian Movements. In Proceedings of the 14th International Symposium on Transportation and Traffic Theory, pp. 235 – 254. Oxford: Elsevier Science, Jerusalem, Israel (1999)

6. Board London Transport: Secord Report of the Operational Research Team on the Capacity of Footways. London Transport Borad – Report, London (1958)

7. Breuer, L., Baum, D.: An Introduction to Queueing Theory: And Matrix-Analytic Methods, Springer (2005)

8. Brown, J.R.: The Knapsack sharing problem. Operation Research. Operation Research, 27(2), 340 – 355 (1979)

9. Bukowski, R.W. Emergency Egress From Buildings. Part 1. History and Current Regulations for Egress Systems Design. In Proceedings of the Conference Sky is the limit, pp. 167 – 191. Society of Fire Protection Engineers, Auckland, New Zealand (2008)

10. Capri, S., Garafolo, C., Ignaccolo, M., Inturri, G., Pluchino, A., Rapisarda, A., Tudisco, S.: Agent-Based Simulation of Pedestrian Behaviour. In Proceedings of the Symposium

on Engineered and Natural Complex Systems-Modeling, Simulation and Analysis, p. 375 (2009)

11. Casadesús Pursals, S., Garriga Garzon, F.: Basic Principle for the Solution of the Building Evacuation Problem. Journal of Industrial Engineering and Management, 2(3), pp. 499 – 516 (2009)

12. Casadesús Pursals, S., Garriga Garzon, F.: Building evacuation: Principles for the analysis of basic structures through dynamic flow networks. Journal of Industrial Engineering and Management, 6(4), pp. 831 – 859 (2013).

13. Chalmet, L.G., Francis, R.L., Saunders, P.B.: Network Models for Building Evacuation, Management Science. 28(1) 86–105 (1982)

14. Cheng, H., Hadjisophocleous, G.V.: Dynamic Modeling of Fire Spread in Building, Fire Safety Journal. 46(4) 211–224 (2011)

15. Choi, W., Hamacher, S., Tufecki, S.: Modeling of building Evacuation Problems by Network flow with Side Constraints. European Journal of Operational Research, 35(1), 98 – 110 (1988)

16. 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)

17. Cova, T.J., Johnson, J.P.: A network flow model for lane-based evacuation routing, Transportation Research Part A: Policy and Practice, 37(7), 579–604 (2003)

18. DeLoach, S.A., Oyenan, W.H., Matson, E.T.: A Capabilities Based Model for Artificial Organizations. Journal of Autonomous Agents and Multi-agent Systems. 16(1), 13–56 (2008)

19. Doheny, J.G. Fraser, J. L.: MOBEDIC - A Decision Modelling Tool For Emergency Situations. Expert Systems with Applications, 10(1), 17 – 27 (1996)

20. 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)

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

22. Dimakis, N., Filippoupolitis, A., Gelenbe, E.: Distributed Building Evacuation Simulator for Smart Emergency Management, The Computer Journal. 53, 1384-1400 (2010)

23. Ebihara, M. ., Ohtsuki, A., Iwaki, H.: Model For Simulating Human BehaviorDuring Emergency Evacuation Based On Classificatory Reasoning And Certainty Value Handling. Shimizu Technical Research Bulletin, 11, 27 – 33 (1992)

24. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In Proceedings of the 3rd International Conference on Multi Agent Systems, pp. 128–135. IEEE Computer Society, Washington, DC (1998)

25. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton, NJ (1962)

26. 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)

27. Francis, R. L., Kisko, T. M.: EVACNET+: A Computer Program to Determine Optimal Building Evacuation Plans. Fire Safety Journal, 9, 211 – 220 (1985).

28. Francis, R.L.: A Uniformity Principle for Evacuation Route Allocation. Journal of Research of National Bureau of Standards, 86(5), 509 – 513 (1981)

29. Francis, R.L.: A Simple Graphical Procedure to Estimate the Minimum time to Evacuate a Building. Society of Fire Protection Engineers – Report (1979)

30. Francis, R.L., Saunders, P.B.: EVACNET: Prototype Network Optimisation Model for Building Evacuation. National Bureau of Standards (1979)

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

32. 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)

33. 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)

34. 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)

35. 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)

36. Friedler, F., Fan, L.T., Imreh, B.: Process Network Synthesis: Problem Definition. Networks. 28, 119–124 (1998)

37. Fruins, J.J.: Pedestrian Planning and Design, Revised Edition. Metropolitan Association of Urban Designers and Environmental Planners, New York, NY, USA (1971)
38. Galea, E.R., Gwinne, S., Lawrence, P., Filipidis, L.: Modeling Occupant Interaction with Fire Conditions using the building EXODUS evacuation Model. Fire Safety Journal, 36(4) 327 – 357 (2001)

39. Garcia-Ojeda, J. C., Bertok, B., Friedler, F., Argoti, A., Fan, L.T. "A Preliminary Study of the Application of the P-graph Methodology for Organization-based Multiagent System Designs: Assessment" *Acta Polytechnica Hungarica* (2015), *To Appear*.

40. García-Ojeda, Juan C., Bertok, Botond, Friedler, Ferenc, Fan, L.T. "On Combining the P-graph Framework and Absorbing Markov Chains for Assessing the Reliability and Cost of Organization-based Multiagent System Design Models," In: *Proceedings of the 6th Veszprem Optimisation Conference: Advanced algorithms* (Veszprem, Hungary). VOCAL 2014.

41. García-Ojeda, J.C., Bertok, B., Friedler, F., Fan, L.T.: Building-evacuation-route planning via time-expanded process-network synthesis. Fire Safety Journal, 61, pp. 338 – 347 (2013)

42. Garcia-Ojeda, J.C., Bertok, B., Friedler, F.: Planning evacuation routes with the P-graph framework, Chemical Engineering Transactions. 29 1531 – 1536 (2012)

43. Garcia-Ojeda, J.C.: On Modeling Building Evacuation Route Plans by Resorting to P-graph, Revista Colombiana de Computación. 12(1), 111–125 (2011)

44. 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)

45. Getachew, T., Kostreva, M., Lancaster, L.: A Generalization of Dynamic for Pareto Optimisation in Dynamic Networks. RAIRO Operation Research, 34, 27 – 47 (2000)

46. Getachew, T.: An Algorithm for Multiple-objective Path Optimisation with Time Dependant Links. In Proceedings of the 10th International Conference on Multicriteria Decision Making, pp. 319 – 330. International Society on Multicriteria Decision Making, Finland (1992)

47. Hamacher, H.W., Tjandra, S.A.: Mathematical Modeling of Evacuation Problems: A state of the art, in: Schreckenberg, M., Sharma, S.D. (eds) Pedestrian and Evacuation Dynamics, pp. 227–266. Springer, Berlin (2002)

48. Hamacher, H.W., Tufekci, S.: On the Use of Lexicographic Min Cost Flows in Evacuation Modeling, Naval Research Logistics, 34 487 – 503 (1987).

49. Han, L., Potter, S., Beckett, G., Pringle, G., Welch, S., Koo, S.H., Gerhard, W., Usmani, A., Torero, J.L., Tate, A.: FireGrid: An e-infrastructure for next-generation emergency response support, J. Parallel Distrib. Comput. 70(11) 1128 – 1141 (2010)

50. Han, L.D., Yuan, F., Urbanik, T.: What is an Effective Evacuation Operation? Journal of Urban Planning and Development. 3–8 (2007)

51. 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)

52. Henderson-Sellers, B., Giorgini, P.: Agent-Oriented Methodologies. Idea group Inc. Hershey, PA (2005)

53. Hope, B., Tardos, E.: The Quickest Transshipment Problem. Journal of Mathematics of Operations Research, 25(1), 36 – 62 (2000)

54. Hope, B. Tardos, E.: Polynomial Time Algorithms for Some Evacuation Problems. In Proceedings of the Fifth annual ACM-SIAM symposium on Discrete algorithms, pp. 433 – 441. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1994)

55. 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)

56. Jin, Y., Levitt, R.E.: The virtual design team: A computational model of project organizations. Computational & Mathematical Organization Theory. 2(3), 171–196 (1996)

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

58. Kim, S., Shekhar, S., Min, M.: Contraflow transportation network reconfiguration for evacuation route planning, IEEE Transactions on Knowledge and Data Engineering. 20(8), 1115–1129 (2008)

59. Kisko, T., Francis, R., Nobel, C.: EVACNET4 User's Guide. University of Florida (1998)

60. Kisko, T.M., Francis, R.L.: Network Models of Building Evacuation. In National Technical Information Service (ed.) Development of Software System, Washington, DC (1984)

61. Kholshenikov, V.V., Shields, T.H., Boyce, K.E., Samoshin, D.A.: Recent Developments in Pedestrian Flow Theory and Research in Russia. Fire Safety Journal, 43, 108 – 118 (2006)

62. Klügl, F., Rindsfuser, G.: Large-Scale Agent-Based Pedestrian Simulation. In Petta, P. et al. (eds.) MATES 2007. LNAI, vol. 4687, pp. 145 – 156 (2007)

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

64. Klupfel, H., Konig, T.M., Wahle, J., Schreckenbe, M.: Microscopic Simulation of Evacuation Processes on Passenger Ships. In Bandini, S., Worsch, T. (eds.) Theory and Practical Issues on Cellular Automata, pp. 63 – 71. Springer-Verlag, London, UK (2000)

65. 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)

66. Kostreva, M.M., Wiecek, M.M., Getachew, T.: Optimization models in fire egress analysis for residential buildings. Fire Safety Science, 3, 805 – 814 (1991)

67. Kretz, T.: Pedestrian Traffic: Simulations and Experiments. Ph.D. Dissertation, University of Duisburg-Essen (2007)

68. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 2, pp. 83 – 97 (1955)

69. Kuhn, H.W.: Variants of the Hungarian method for assignment problems. Naval Research Logistics Quarterly. 3, pp. 253 – 258 (1956)

70. Kuligowski, E.D., Peacock, R.D., Hoskins, B.L.: A Review of Building Evacuation Models NIST, Fire Research Division, 2nd edition. Technical Note 1680 Washington, US (2010)

71. Kuligowski, E.D., Mileti, D.S.: Bibliography on Evacuation From Building Fires: Education, Behavior and Simulation Techniques. National Institute of Standards and Technology – Report (2007)

72. Kuligowski, E.D., Peacock, R.D.: A Review of Building Evacuation Models NIST, National. Technical Note 1471 (2005)

73. Lovas, G.G.: Models of Way Finding in Emergency Evacuations. European Journal of Operation Research, 105, 371 – 389 (1998)

74. Lu, Q., George, B., Shekhar, S.: Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. In Bauzer, C. et al. (eds.) SSTD 2005, LNCS 3633, pp. 291 – 307. Springer-Verlag, Berlin Heidelberg (2005)

75. Lu, Q., Huang, Y., Shekhar, S.: Evacuation Planning: A Capacity Constrained Routing Approach. In Chen, H. et al. (eds.) ISI 2003, LNCS 2665, pp. 291 – 307. Springer-Verlag, Berlin Heidelberg (2003)

76. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink, (2005)

77. Miah, M.: Survey of Data Mining Methods in Emergency Evacuation Planning. In Proceedings of the Conference for information Systems Applied Research, 4(1815). Education Special Interest Group of the Association of Information Technology Professionals, Chicago, IL (2011)

78. Minoux, M.: Networks synthesis and optimum network design problems: Models, solution methods and applications. Networks, 19, 313–360 (1989)

79. Munkres, J.: Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics. 5(1), pp. 32 – 38 (1957)

80. Nagel, K., Schreckenberg, M. A.: A Cellular Automaton Model for Freeway Traffic. Journal of Physique I, 2, 2221 – 2229 (1999)

81. 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)

82. Nelson, H.E., McLennan, H.A.: Emergency Movement. In National Fire Protection Association (ed.) The SPE Handbook of Fire Protection Engineering, 3, pp. 286 – 295 (1996)

83. 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)

84. Pauls, J.L.: Movement of People. In National Fire Protection Association (ed.) The SPE Handbook of Fire Protection Engineering, 3, pp. 263 – 285 (1996)

85. Pauls, J. L.: The Movement of People in buildings and Design Solutions for Means of Egress. Fire Technology, 20(1), 27 – 47 (1984)

86. Pauls, J. L., Jones, B.K.: Building Evacuation: Research Methods and Case Studies. In Canter, D. (ed.) Fires and Human Behavior, pp. 227 – 251 (1980)

87. Peacock, R.D., Kuligowski, E.D., Averill, J. D.: Pedestrian and Evacuation Dynamics, Springer-Verlag (2011)

88. Picard, G., Mellouli, S., and Gleizes, M.: Techniques for Multi-agent System Reorganization. In: Dikenelli, O. et al. (eds.) ESAW 2005, LNAI 3963, pp. 142–152. Springer-Verlag, Berlin Heidelberg (2006)

89. Poon, L. S.: Evacsim: A Simulation Model Of Occupants With Behavioural Attributes in Emergency Evacuation Of High-rise Building Fires. Fire Safety Science, 4, 681–692 (1994)

90. Predtechenskii, V. M., Milinskii, A.I.: Planning Foot Traffic Flow in Buildings. Amerind Publishing Co, New Delhi, India (1978)

91. Proulx, G.., Kaufman, A., Pineau, J.: Evacuation Time and Movement in office Buildings. National Research Council Canada – Report (1996)

92. Pu, S., Zlatanova, S.: Evacuation Route Calculation of Inner Buildings, in: van Oosterom, P. et al. (eds.) Geo-information for Disaster Management, pp. 1143–1161. Springer, Berlin (2005)

93. P-graph – PNS studio, http://www.p-graph.com

94. Qingsong, L., Betsy, G., Shashi, S.: Capacity constrained routing algorithms for evacuation planning: a summary of results. In: Bauzer, C. et al. (eds.) SSTD 2005, LNCS 3633, pp. 291 – 307. Springer-Verlag, Berlin Heidelberg (2005)

95. 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)

96. Ronald, N., Sterling, L., Kirley, M.: An Agent-based Approach to Modeling Pedestrian Behavior. International Journal of Simulation Systems, Science & Technology, 8(1), 25 – 38 (2007)

97. Sangho, K., Shashi, S.: Contraflow network reconfiguration for evacuation planning: a summary of results. In Proceedings of the 13th annual ACM international workshop on Geographic information systems (GIS '05), pp. 250 – 259. ACM, New York, NY (2005)

98. Serugendo, G.D.M, Gleizes, M.P., Karageorgos, A.: Self-organisation and emergence in mas: An overview. Informatica 30(1), 45–54 (2006)

99. Sims, M., Corkill, D., Lesser, V.: Automated organization design for multi-agent systems. Auton. Agents and Multi-Agent Syst 16(2), 151–185 (2008)

100. Skutella, M.: An Introduction to Network Flows Over Time. In Cook, W. et al. (eds) Research Trends in Combinatorial Optimization, pp. 451–482. Springer, Berlin (2009)

101. Smidla, J. and I. Heckl, S-graph Based Scheduling Method for Parallel Architecture, Chemical Engineering Transactions, 21, 937-942 (2010)

102. Smith, J.M., Bakuli, D.: Resource Allocation in state Dependent Emergency Evacuation Networks. European Journal of Operation Research, 89(3), 543 – 555 (1996)

103. Smith, J.M., Talebi, K.: Stochastic Network Evacuation Models. Computer & Operation Research, 12(6), 559 – 577 (1985)

104. Smith, J.M., Karbowicz, C.J.: A K-shortest Paths Routing Heuristic for Stochastic Network Evacuation Models. Engineering Optimization, 7, 253 – 280 (1984)

105. Stringfield, W.H.: Emergency planning and management, first ed. Government Institutes, Rockville, MD (1996)

106. Timmermans, H.J.P.: Pedestrian Behavior: Models, Data Collection, and Applications. Emerlad Group Publishing, London, UK (2009)

107. Togawa, K.: Study of Fire Escapes Based on the Observation Multitude currents. Japan Building Research Institute – Report (1955)

108. Thompson, P.A., Marchant, E.W.: Testing and Application of the Computer Model SIMULEX. Fire Safety Journal, 24, 149 – 166 (1995)

109. Thompson, P.A., Marchant, E.W.: A Computer Model for the Evacuation of Large Building Populations. Fire Safety Journal, 24, 131 – 148 (1995)

110. Tsai, J., Fridman, N., Bowring, E., Brown, M., Epstein, S., Kaminka, G., Marsella, S., Ogden, A., Rika, I., Sheel, A., Taylor, M., Wang1y, X., Zilka, A., Tambe, M.: ESCAPES: evacuation simulation with children, authorities, parents, emotions, and social comparison. In Sonnenberg, L. (eds.) AAMAS 2011, pp. 457 – 464. International Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan (2011)

111. Varga, J. B., Friedler, F., Fan, L. T., Parallelization of the Accelerated Branch-and-Bound Algorithm of Process Synthesis: Application in Total Flowsheet Synthesis, Acta Chimica Slovenica, 42, 15-20 (1995)

112. Wiecek, T. Approximation in Time-dependent Multi-objective Path Planning. In Proceedings of the Conference on Systems, Man, and Cybernetics, pp. 861 – 866. IEEE Computer Society, Chicago, IL, USA (1992)

113. Wiecek, T. Multicriteria Decision Making in Fire Egress Analysis. In Proceedings of the IFAC/IFORS Workshop on Support Systems for Decision and Negotiation, pp. 285-290. System Research Institute, Warsaw, Poland (1992)

114. Zambonelli, F., Jennings, N.R., Wooldridge, M.J.: Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems. International Journal of Software engineering and Knowledge Engineering. 11(3), pp. 303 – 328 (2001)

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

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

Appendix A. Process-Network Synthesis (PNS)

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 also be 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 [1].

A number of methods has been developed for process synthesis [1]. 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 [1]. 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 [2]. 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 [3], SSG [3], and ABB [4,5].

References

1 Friedler, F., Fan, L.T., Imreh, B.: Process Network Synthesis: Problem Definition, Networks, 28, 119 – 124 (1998)

2 Minoux, M.: Networks synthesis and optimum network design problems: Models, solution methods and applications, Networks, 19, 313 – 360 (1989)

3 Friedler, F., Tarján, K., Huang, Y.W., Fan, L.T.: Graph-theoretic approach to process synthesis: axioms and theorems, Chem. Engng. Sci., 47, 1972 – 1988 (1992)

4 Friedler, F., Tarján, K., Huang, Y.W., Fan, L.T., Combinatorial Algorithms for Process Synthesis, Computers Chem. Engng., 16, S313 – 320 (1992)

5 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)

Appendix B. Process Graph (P-graph)

The mathematical definition of a P-graph and a process structure represented by it are elaborated below [1].

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

$$O \subseteq \wp(M)\ x\ \wp(M) \tag{B.1}$$

Thus, a P-graph can be defined to be a pair, (M,O), as follows: the vertices of the graph are the elements of

$$V = M \times O \tag{B.2}$$

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

$$A = A_1 \cup A_2 \tag{B.3}$$

where

$$A_1 = \{(X,Y) | Y = (\alpha,\beta) \in O \text{ } and \text{ } X \in \alpha\} \quad \text{(B.4)}$$

and

$$A_2 = \{(Y,X) \mid Y = (\alpha,\beta) \in O \text{ } and \text{ } X \in \beta\} \quad \text{(B.5)}$$

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. The arcs between the nodes signifiy that a material is input to or ouput from an operating unit. Hence, P-graphs are bipartite graphs as mentioned earlier.

Also, the P-graph representation of a process network should observe the constrains imposed by the process itself [2]. For instance, the maximum available raw materials may be constrained, and the rate of manufacturing of each product must be specified. An operating unit produces its output materials if all its input materials are supplied. The input materials are consumed according to the rates given on the arcs leading to the respective operating unit. The input and output materials, and the aforementioned rates collectively define formally an operating unit. Moreover, an operating unit may have upper and lower capacities. At any material node, the sum of the outgoing flows is equal to the sum of the incoming flows, i.e., the mass balance holds.

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 $O = \{(\{B\},\{A\}),(\{D,E\},\{B,C\}),(\{F\},\{A,C\}),(\{F\},\{A,C\})\}$. It is not difficult to validate that sets M and O satisfies constraint (B.1), i.e., (M,O) is a P-graph, as depicted in Figure B.1.

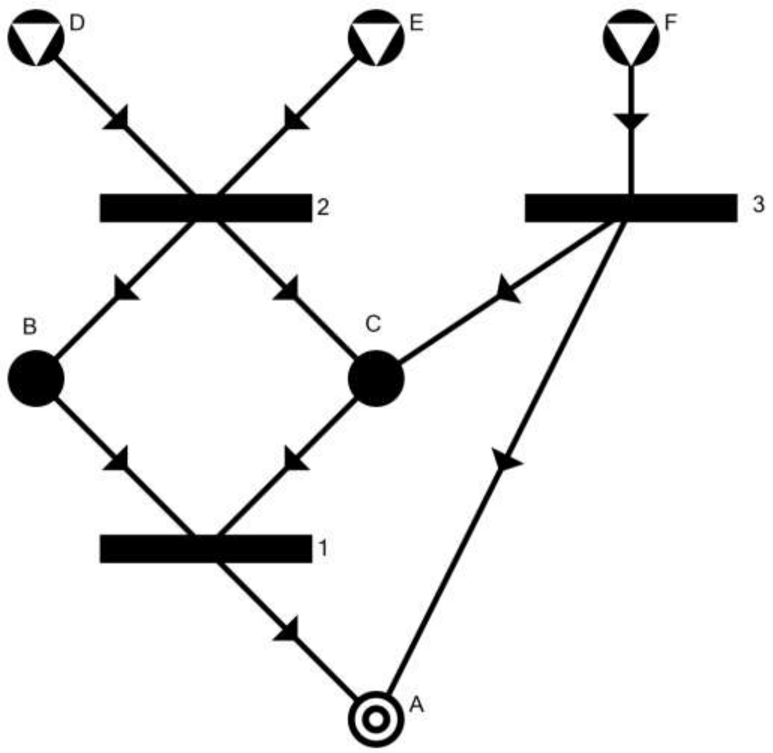


Figure B.1. P-graph (*M*,*O*) where *A*,*B*,*C*,*D*,*E*, and *F* are materials, and *1*,*2*, and *3* are the operating units: ▽ represents raw materials or input elements of the whole process; ● symbolizes intermediate-materials or elements, emerging between the operating units; and ◎ represents products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

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

(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.

If a P-graph of a given synthesis problem, (P, R, O)[27], satisfies theses axioms, it is defined to be a solution-structure of the problem. For example, Figure B.2 depicts an example of two solution-structures for synthesis problem (P_1, R_1, O_1) with

$M_1 = \{A, B, C, D, E, F, G, H, I\}$

$P_1 = \{A\}$

$R_1 = \{D, F, H\}$

[27] 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.

and

$$O_1 = \{(\{C\},\{A,I\}),(\{B\},\{A,E\}),(\{D,E\},\{B\}),(\{E,F\},\{B\}),(\{F,G\},\{C\}), \{H,I\},\{G\})\}.$$

Note that a solution-structure does not necessarily contain all the components defined in the set of materials, e.g., M_1; neither does it necessarily utilize all the components specified in the set of raw materials, e.g., R_1.

Since the final product, A, is presented as an M-type vertex in both Figure B.2 (a) and (b), axiom (S1) is satisfied by the solution-structures depicted in these figures. Axiom (S2) is satisfied in that vertex F in Figure B.2 (a) and vertices F and H in Figure B.2 (b) are the only vertices without an input; they represent raw materials. Figure B.2 (a) contains two operating units, $(\{E,F\},\{B\})$ and $(\{B\},\{A,E\})$, and Figure B.2 (b) contains three operating units $(\{C\},\{A,I\})$, $(\{F,G\},\{C\})$, and $(\{H,I\},\{G\})$; all these operating units are defined in the synthesis problem, thereby satisfying axiom (S3). In conformity with axiom (S4), every vertex of the type O-type in either Figure B.2 (a) or (b) does have at least one path leading to vertex A representing the final product. For example, the path in Figure B.2 (a), comprising three arcs, namely, $((\{E,F\},\{B\}),B),(B,(\{B\},\{A,E\}))$, and $((\{B\},\{A,E\}),A)$, links vertex $(\{E,F\},\{B\})$, representing an operating unit, to vertex A which is the final product. Axiom (S5) is satisfied by virtue of the fact that every vertex of the M-type belonging to the graph of either Figure B.2 (a) or (b) is an input to or output from at least one vertex of the O-type in the respective graph.

Thus all axioms are satisfied by the structures in Figure B.2 (a) or (b). As counterexample, Figure B.3 illustrates a P-graph that is not a solution structure of synthesis problem (P_1, R_1, O_1), because axioms (S1), (S2), (S4), and (S5) are not satisfied.

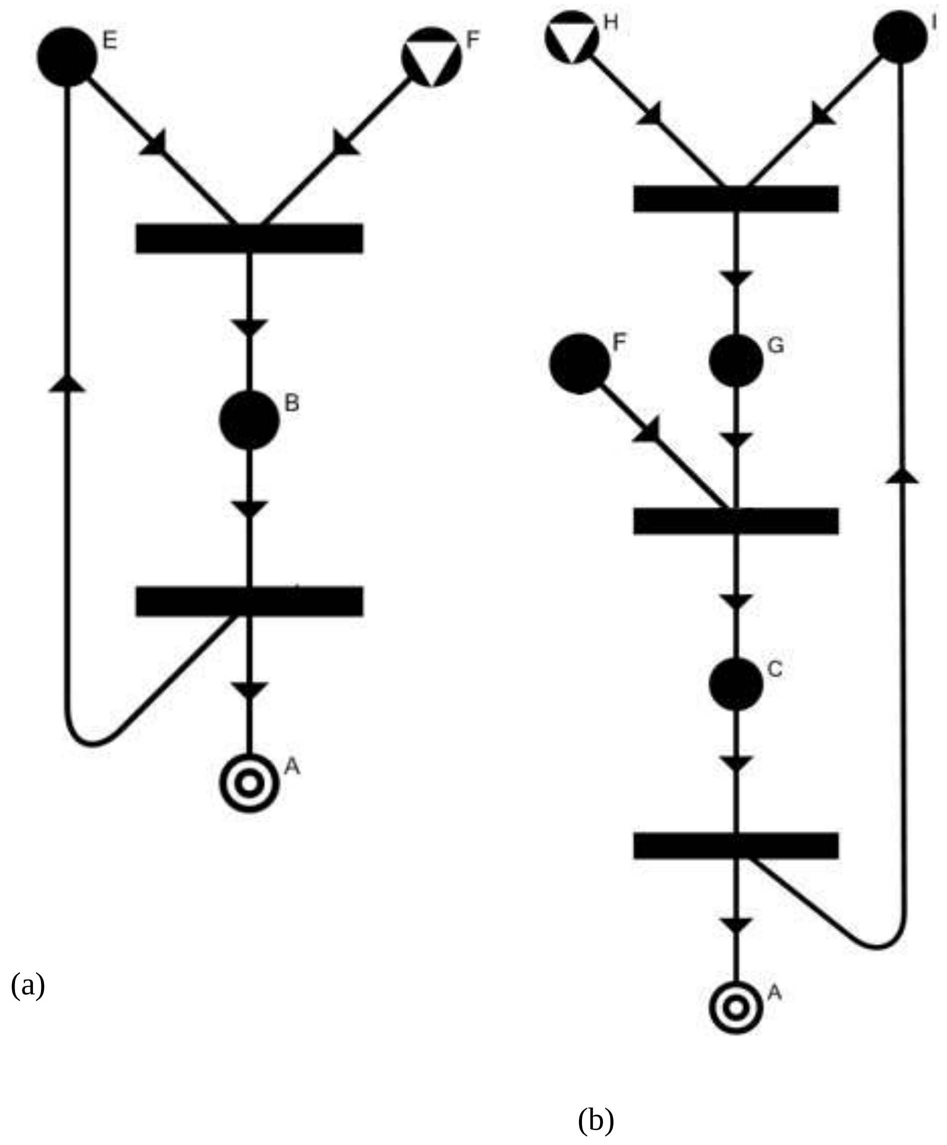


Figure B.2 Two solution-structures for the synthesis problem (*P_1*, *R_1*, *O_1*).

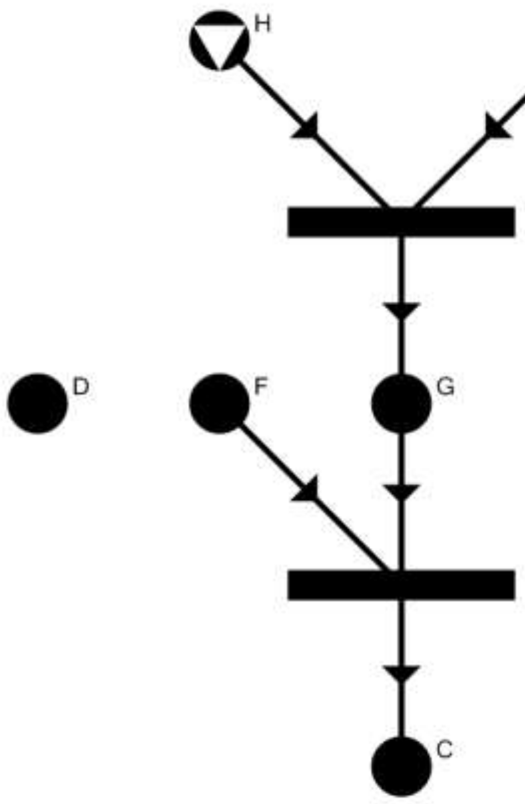


Figure B.3. P-graph that is not a solution-structure for synthesis problem (*P_1, R_1, O_1*).

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 [1,3,4,5].

References

1 Friedler, F., Tarján, K., Huang, Y.W., Fan, L.T.: Graph-theoretic approach to process synthesis: axioms and theorems, Chem. Engng. Sci., 47, 1972 – 1988 (1992)

2 Friedler, F., Fan, L.T., Imreh, B.: Process Network Synthesis: Problem Definition, Networks, 28, 119 – 124 (1998)

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

4 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)

5 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)

Appendix C. Short Summary of Pidgin Algol

Pidgin Algol is a high-level language whose purpose is to describe algorithms for publication and mathematical examination [1,2]. This language uses traditional mathematical and programming language constructs, such as expressions, conditions, statements, and procedures. It does not have a fixed set of data types.

Statements

```
variable≔ expression;
if condition then statement else statement;
while condition do statement;
repeat statement until condition;
for variable≔initial-value step step-size until final-value do statement;
for all $x \in X$ do statement;
label:statement;
goto label;
begin
        statement;
        statement;
…
        statement;
end;
procedure name (list of parameters): statement
return;
return expression;
comment comment;
any othe miscelaneous statements
```

References

1 Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA (1974)

2 Papadimitriou, C.H.: Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, NJ (1982)

Appendix D. Markov Chains

In probability theory, a Markov chain or Markov model is a special type of discrete stochastic process in which the probability of an event occurring depends on the immediately preceding one (i.e., the next event depends only on the current state and not on the sequence of events that preceded it). This feature distinguishes Markov chains of independent events (e.g., tossing a coin or rolling a die). Markov chains get their name after Andrei Markov, Russian mathematician (1856 – 1922), in 1907.

Specifying a General Markov Chain

A Markov chain is formally described as follows. Let $\{X_n \mid n \in \mathbb{N} \; and \; n > 0\}$ be a stochastic process, in discrete time, with finite or infinite state space S is a Markov chain with stationary transition probabilities if it satisfies:

For each $n \geq 1$, if A is an event depending on any subset of $\{X_n, X_{n-1}, X_{n-2}, \dots, X_1\}$, then, for any states i and j in S,

$$P_{markov}(X_{n+1} = j \mid X_n = i \wedge A) = P(X_{n+1} = j \mid X_n = i) \qquad \text{(D.1)}$$

For any given states i and j

$$P_{markov}(X_{n+1} = j \mid X_n = i) \text{ holds } \forall n \geq 1, \qquad \text{(D.2)}$$

where, Eq. D.1 is the Markov property. More generally, for each $n \geq 1$ and $m \geq 1$, if A (as defined in Eq. D.1), then for any states i and j in S:

$$P_{markov}(X_{n+m} = j \mid X_n = i \wedge A) = P_{markov}(X_{n+m} = j \mid X_n = i), \qquad \text{(D.3)}$$

denotes transition probabilities in Eq. D.2 by

$$p_{ij} = P_{markov}(X_{n+1} = j \mid X_n = i\,). \qquad (D.4)$$

The Transition Matrix *P*

The transition matrix P for a Markov chain with state space $S = \{s_1, s_2, \dots, s_n\}$, where $n \in \mathbb{N}$, and one-step transition probabilities p_{ij} is the $n \times n$ matrix.

$$P_{markov} \overset{\text{def}}{=} \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}$$

Note that the matrix P satisfies

$$0 \le P_{markov_{ij}} \le 1, \quad 1 \le i, j, \le n \qquad (D.5)$$

$$\textstyle\sum_{j=1}^{N} P_{markov_{ij}} = 1, \quad 1 \le i \le n \qquad (D.6)$$

Example

Example 1: The Veszprém weather (adapted from [2]). It is sometimes claimed that the best way to predict tomorrow´s weather is simply to guess that it will be the same tomorrow as it is today. If we assume that this is correct, then is it natural to model the weather as a Markov chain. For the sake of simplicity, we assume the there are five kinds of weather: snowy, cloudy, rainy, sunny, and windy. Then the weather forms a Markov chain with state space $\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, \mathrm{s}_4, \mathrm{s}_5\}$ (with $\mathrm{s}_1 =$ "snowy", $\mathrm{s}_2 =$ "cloudy", $\mathrm{s}_3 =$ "rainy", $\mathrm{s}_4 =$ "sunny", and $\mathrm{s}_5 =$ "sunshine") and the transition matrix[28].

[28] Transition probabilities, p_{ij}, are just hypothetical and used for illustration.

$$P_{markov} = \begin{pmatrix} 0.5 & 0.25 & 0.05 & 0.045 & 0.155 \\ 0.125 & 0.65 & 0.025 & 0.15 & 0.05 \\ 0.075 & 0.45 & 0.125 & 0.15 & 0.2 \\ 0.025 & 0.1 & 0.1 & 0.75 & 0.025 \\ 0.1 & 0.05 & 0.6 & 0.05 & 0.2 \end{pmatrix}$$

Graphical Description

A useful way to picture a Markov chain is its so-called transition graph. The transition graph consists of nodes representing the states of the Markov chain, and arrows between the nodes, representing the transition probabilities. This is explained by showing the transition graph of the example considered above (see Figure D.1).

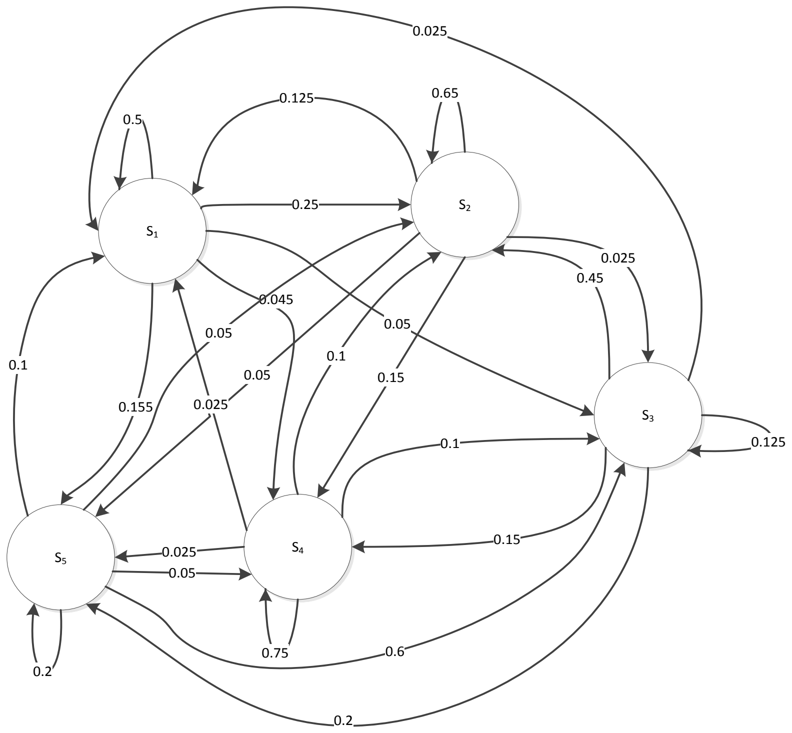


Figure D.1. Transition graph for the Veszprem weather example.

Absorbing Markov Chain

An Absorbing Markov chain is a type of Markov chain in which every state can reach an absorbing state. An absorbing state is a state that, once entered, cannot be left [1].

Specifying an Absorbing Markov Chain

An Absorbing Markov chain is formally described in Eqs. D.1 – D.4, with the particularity that:

a. A state $s_i \in S$ is called absorbing if it is impossible to leaving it (i.e., $p_{ii} = 1$)
b. A Markov is absorbing if it has at least one absorbing state and if from every state it is possible reach an absorbing state (not necessarily in one step).
c. In an absorbing Markov chain, a state which is not absorbing is called *transient*.

Example

Example 2: The Drunkard's Walk: A man walks along a four-block stretch of Park Avenue. If he is at corner 1, 2, or 3, then he walks to the left or right with equal probability. He continues until he reaches corner 4, which is a bar, or corner 0, which is his home. If he reaches either home or the bar, he stays there. Then each man's decision, i.e., walk to the left or walk to the right, on each corner forms a Markov chain with state space $S = \{s_1, s_2, s_3, s_4, s_5\}$ (with $s_1 = "corner\ 0"$, $s_2 = "corner\ 1"$, $s_3 = "corner\ 2"$, $s_4 = "corner\ 3"$, and $s_5 = "corner\ 4"$) and the transition matrix (see Figure D.2).

$$P_{markov} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

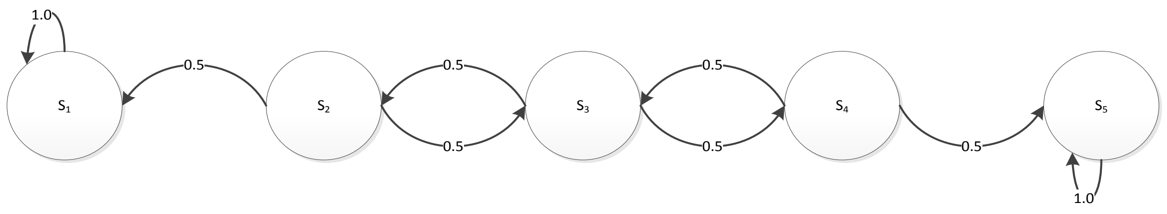


Figure D.2. Transition graph for the Drunkard's Walk example.

The Canonical Form of a transition matrix *P* representing an Absorbing Markov Chain

By permuting the states of an absorbing chain so that the transient states come first, we can write the transition matrix of the absorbing chain as

$$P_{markov} = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix} \tag{D.7}$$

The matrix Q describes the transition probabilities between transient states, R the transition probabilities from transient to absorbing states (R should not be the matrix of all zeros), I is the identity matrix since the chain stays at absorbing states, and, 0 is the zero matrix [3].

The fundamental matrix, F, for an absorbing Markov chain is defined as follows:

$$F = (I_n - Q)^{-1} \tag{D.8}$$

Where, I_n is the $n \times n$ identity matrix corresponding in size to matrix Q, so that the $I_n - Q$ exists.

For the Drunkard's Walk example, using I_3 gives

$$F = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix} \right)^{-1}$$

$$F = \left(\begin{bmatrix} 1 & -0.5 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -0.5 & 1 \end{bmatrix}\right)^{-1}$$

$$F = \begin{bmatrix} 1.5 & 1 & 0.5 \\ 1 & 2 & 1 \\ 0.5 & 1 & 1.5 \end{bmatrix}$$

The fundamental matrix, F, gives the expected number of visits to each state before absorption occurs. For example, if the man is currently in $corner$ 1, i.e., s_2, the first row of F says that he expects to have 1.5 time periods on average in this state and 1.5 + 1 + 0.5 = 3 time periods in the various *transient* states before reaching the bar or home.

To see why this is true, consider an absorbing Markov chain currently in state i. The expected number of times that the chain visits state j at this step is 1 for i and 0 for all other states. The expected number of times that the chain visits state j at the next state is given by the element row i, column j of the transition matrix Q. The expected number of times the chain visits state j two steps from now is given by the corresponding entry in the matrix Q^2. The expected number of visits in all steps is given by

$$I + Q + Q^2 + Q^3 + \cdots \tag{D.9}$$

To find out whether this infinite sum is the same as

$$(I - Q)^{-1}, \tag{D.10}$$

multiply Eq. D.9 by Eq. D.10, thus

$$(I + Q + Q^2 + Q^3 + \cdots)(I - Q)^{-1} = I + Q + Q^2 + Q^3 + \cdots - Q - Q^2 - Q^3 = I, \tag{D.11}$$

which verifies our result.

It can be shown that

$$ {P_{markov}}^{k} = \begin{bmatrix} I_m & 0 \\ (I + Q + Q^2 + Q^3 + \cdots + Q^{k-1})R & Q^k \end{bmatrix}, \tag{D.12} $$

where I_m is the $m \times m$ identity matrix. As $k \to \infty$, $Q^k \to O_n$, the $n \times n$ zero matrix, and

$$ {P_{markov}}^{k} = \begin{bmatrix} I_m & 0 \\ FR & 0_n \end{bmatrix}, \tag{D.13} $$

So we see the FR gives the probability that if the systems was originally in a non-absorbing state, it ends up in one of the absorbing states.Finally, use the fundamental matrix F along with matrix R (see Eq. D.7) to get the product FR.

$$ FR = \begin{bmatrix} 1.5 & 1 & 0.5 \\ 1 & 2 & 1 \\ 0.5 & 1 & 1.5 \end{bmatrix} * \begin{bmatrix} 0.5 & 0 \\ 0 & 0 \\ 0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.75 & 0.25 \\ 0.5 & 0.5 \\ 0.25 & 0.25 \end{bmatrix} $$

The product matrix FR gives the probability that if the system was originally in a particular non-absorbing state; it ended up in the absorbing state. For example, the probability is 0.75 that if the man was originally in $corner$ 1, i.e., s_2, he ended up at home.

The Power Method

However the method introduced before might be inefficient for numerically solving large markov chains, including absorbing ones [5]. To deal with that we compute $x_n^{(0)}$ (the stable distribution or a steady state of P that satisfies $x_n^{(k+1)} = x_n^{(k)} * P_{markov}$) by means of an iterative method called the power method. More specifically, given a (stochastic) transition matrix P_{markov}, and an initial vector $x_n^{(0)}$, we compute iteratively $x_n^{(k+1)}$=$x_n^{(k)} * P_{markov}$ until the difference (in some norm) between $x_n^{(k+1)}$ and $x_n^{(k)}$ is small enough (see Figure D.3).

```
input: $P_{markov}, x_n^{(0)}, \varepsilon$
comment: $P_{markov}$ $is$ a transition matrix of an absorbing markov chain, $x_n^{(0)}$ is a
starting vector describing the stable distribution or a stady state of $P$, and $\varepsilon$ is the
accuracy.
output: the stady state of $P_{markov}, x_n^{(k)}$, after $k$ iterations
begin
st1:   $k := 0$;
st2:   $\delta := 0$;
loop1:   while $\delta < \varepsilon$ do
           begin
               $x_n^{(k+1)} = x_n^{(k)} * P_{markov}$;
               $\delta = \left\| x_n^{(k+1)} - x_n^{(k)} \right\|_i$; comment: where $1 \le i \le n$
               $k = k + 1$;
         end;
end;
```

Figure D.3. Algorithm *PowerMethod* written in Pidgin Algol (see Appendix C, adapted from [4]).

For instance, suppose Tables D.1 – D.3 describes the initial location of man in Example 2. In particular, each table shows the initial probability vector, $x_n^{(0)}$, for each case where the man starts walking either on $Corner$ 1, $Corner$ 2, and $Corner$ 3; respectively.

Table D.1. Initial probability vector, $x_n^{(0)}$, for case where the man starts walking on *Corner 1*.

Corner	State	Proportion (100%)
0	s_1 (absorbing)	0
1	s_2	1
2	s_3	0
3	s_4	0
4	s_5	0

Table D.2. Initial probability vector, $x_n^{(0)}$, for case where the man starts walking on *Corner 2*.

Corner	State	Proportion (100%)
0	s_1 (absorbing)	0
1	s_2	0
2	s_3	1
3	s_4	0
4	s_5	0

Table D.3. Initial probability vector, $x_n^{(0)}$, for case where the man starts walking on *Corner 3*.

Corner	State	Proportion (100%)
0	s_1 (absorbing)	0
1	s_2	0
2	s_3	0
3	s_4	1
4	s_5	0

By employing the power method algorithm, presented in Figure D.3, we can compute the probability distribution of the man in the different corners after k iterations as illustrated in Tables D.4 – D.6. Note that the results presented in Tables D.4 – D.6 are equivalent to those computed by employing the fundamental matrix F along with matrix R (see Eq. D.7) to get the product FR.

Table D.4. Probability distribution, after 54 iterations, given the initial probability vector, $x_n^{(0)}$, [0.0, 1.0, 0.0, 0.0, 0,0]. That is, the man starts walking on *Corner 1*.

Iteration, k	Corner 0	Corner 1	Corner 2	Corner 3	Corner 4
1	0,5	0	0,5	0	0
2	0,5	0,25	0	0,25	0
…	…	…	…	…	…
13	0,74609375	0	0,0078125	0	0,24609375

…	…	…	…	…	…
25	0,74993896	0	0,00012207	0	0,24993896
…	…	…	…	…	…
33	0,74999619	0	0	0	0,24999619
…	…	…	…	…	…
54	**0,75**	**0**	**0**	**0**	**0,25**

Table D.5. Probability distribution, after 54 iterations, given the initial probability vector, $x_n^{(0)}$, [0.0, 0.0, 1.0, 0.0, 0,0]. That is, the man starts walking on *Corner 2*.

Iteration, *k*	Corner 0	Corner 1	Corner 2	Corner 3	Corner 4
1	0	0	0,5	0	0,5
2	0	0,25	0	0,25	0,5
…	…	…	…	…	…
13	0,24609375	0	0,0078125	0	0,74609375
…	…	…	…	…	…
25	0,24993896	0	0,00012207	0	0,74993896
…	…	…	…	…	…
33	0,24999619	0	0	0	0,74999619
…	…	…	…	…	…
54	**0,25**	**0**	**0**	**0**	**0,75**

Table D.6. Probability distribution, after 54 iterations, given the initial probability vector, $x_n^{(0)}$, [0.0, 0.0, 0.0, 1.0, 0,0]. That is, the man starts walking on *Corner 3*.

Iteration, *k*	Corner 0	Corner 1	Corner 2	Corner 3	Corner 4
1	0	0	0,5	0	0,5
2	0	0,25	0	0,25	0,5
…	…	…	…	…	…
13	0,24609375	0	0,0078125	0	0,74609375
…	…	…	…	…	…
25	0,24993896	0	0,00012207	0	0,74993896
…	…	…	…	…	…

33	0,24999619	0	0	0	0,74999619
…	…	…	…	…	…
54	**0,25**	**0**	**0**	**0**	**0,75**

References

1 Grinstead, C.M., Snell, J.L. Markov Chains. Introduction to Probability (pp. 405–470). American Mathematical Society. Providence, RI, USA (1997).

2 Häggström, O.: Finite Markov Chains and Algorithmic Applications. Part of London Mathematical Society Student Texts, p. 16, UK (2002)

3 Lial, M.L., Greenwell, R.N., Ritchey, N.P.: Finite Mathematics (10th edition). Pearson. Upper Saddle River, NJ, USA (2011).

4 Nemirovsky, D.: Web graph and PageRank algorithm. In: Jain, R. K. (ed.) Internet Search Engines: An Introduction. ICFAI University Press, India (2007)

5 Panju, M.: Iterative Methods for Computing Eigenvalues and Eigenvectors, The Waterloo Mathematics Review, 1, 9 – 18 (2011)

Appendix E. Series and Parallel Systems Engineering

It is believed that the Second World War propitiated the development of the study of reliability because the equipment reliability problems [1]. One of the first engineers to dig into reliability research was the German rocket engineer Wernher Von Braun (1912 – 1977). Von Braun and his collaborators, during the Second World War, adopted ideas stemming from mechanical reliability to diagnose and fix their V-1 combat rocket, which was infested with reliability problems. Von Braun assumed that if the weakest component of the rocket is fixed, then the rocket would not fail. However, when Von Braun and his colleagues build the least reliable component part more reliable, they found out that the V-1 was still 100% unreliable [4]. Nevertheless, Erich Pieruschka, a German mathematician (1914 – 2004), working with Von Braun on a different project, was invited to express his thoughts about this issue. Pieruschka pointed out that the rocket's reliability was equal to the product of the reliability of its components and not simply to the reliability of the weakest component. This idea was the basis of the modern predictive reliability model [1]. As result, this theory formed the basis for what later became known as Lusser's law after Robert Lusser, German engineer (1889 – 1969), in 1953.

Some Useful Definitions

Before discussing the probabilistic reasoning behind reliability block formulas for series and parallel systems and presenting examples of practical ways of using them, some notations and definitions need to be introduced [4].

The Distribution Function

The distribution function is also often called cumulative distribution function (abbreviated as CDF). Formally, a cumulative distribution function is defined as follows, if X is a random variable, its distribution function is a function $F_X(x): \mathbb{R} \rightarrow [0,1]$ such that

$$F_X(x) = P(X \leq x) \; \forall x \in \mathbb{R} \tag{E.1}$$

where $P(X \leq x)$ is the probability that X is less than or equal to x.

Continuous Random Variable

A continuous random variable is a random variable whose cumulative distribution function is a continuous function. The following is a formal definition of a continuous random variable. A random variable is said to be absolutely continuous if the probability that it assumes a value in a given integral $[a, b]$ can be expressed as an integral:

$$P(X \in [a, b]) = \int_a^b f_X(x)dx \tag{E.2}$$

Where the integral function Eq. E.3 is called the probability density function of X.

$$f_X(x): \mathbb{R} \rightarrow [0, \infty) \tag{E.3}$$

As consequence of this definition, the cumulative distribution function of X is

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)dt \tag{E.4}$$

Properties of Probability Density Function

Probability density functions (usually called PDF) are characterized by two properties. Also, any function that satisfies these two properties is a legitimate PDF. The following proposition formally describes the two properties.

Proposition: Let X be a continuous random variable. Its probability density function, i.e., $f_X(x)$, satisfies the following two properties:

$$f_X(x) \geq 0 \text{ for any } x \in \mathbb{R} \tag{E.5}$$

$$\int_{-\infty}^{\infty} f_X(x)dx = 1 \tag{E.6}$$

Eq. E.5, first property, states that for a function to be a PDF, it must be nonnegative. This makes intuitive sense since probabilities are always nonnegative numbers. Also, Eq. E.6, second property states that the area between $f_X(x)$ and the x-axis must be 1, or that all probabilities must integrate to 1.

Proof: By Eq. E.5, probabilities cannot be negative; therefore Eq. E.2 can be rewritten as

$$\int_{a}^{b} f_x(x)dx \geq 0 \tag{E.7}$$

for any interval $[a, b]$. But the above integral can be non-negative for all intervals $[a, b]$ only if itself is non-negative, i.e., if $f_X(x) \geq 0$ for all x. This proves property 1 above (non-negativity). Furthermore, the probability of a sure thing must be equal to 1. Since $\{X \in (-\infty, \infty)\}$ is a sure thing [2], then

$$1 = P\big(X \in (-\infty, \infty)\big) = \int_{-\infty}^{\infty} f_X(x)dx, \tag{E.8}$$

which proves Eq. E.6.

Exponential Distribution

The exponential distribution is defined as follows. Let X be a continuous random variable. Let its support, i.e., the set of values that X can take, be the set of positive real numbers, $\mathbb{R}^{\geq 0}$. Let $\lambda \in \mathbb{R}^{>0}$. We say that X has an exponential distribution with parameter λ (called the rate parameter) if its probability density function is

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & if\ x \in\ \mathbb{R}^{\geq 0} \\ 0 & if\ x \notin\ \mathbb{R}^{\geq 0} \end{cases} \tag{E.9}$$

A random variable having an exponential distribution is also called an exponential random variable. The following is a proof that $f_{X(x)}$ is a probability density function.

Proof: Proving Eq. E.5 is obvious. We need to prove Eq. E.6. This is proved as follows:

$$\begin{aligned} \int_{-\infty}^{\infty} f_X(x)dx &= \int_{0}^{\infty} \lambda(e^{-\lambda x})dx \\ &= -e^{-\lambda x}|_0^\infty \\ &= 0-(-1) \\ &= 1 \qquad \blacksquare \end{aligned}$$

Cumulative Distribution Function of the Exponential Distribution

The distribution function of an exponential random variable, X, is:

$$F_X(x) = \begin{cases} 0 & if\ x < 0 \\ 1-e^{-\lambda x} & if\ x \geq 0 \end{cases} \tag{E.10}$$

Proof: if $x < 0$, then:

$$F_X(x) = P(X \leq x) = 0 \tag{E.11}$$

Because X cannot take on negative values. If $x \geq 0$, then:

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= \int_{-\infty}^{x} f_X(t)dt \end{aligned}$$

$$= \int_{0}^{x} \lambda(e^{-\lambda t})dt$$
$$= -e^{-\lambda t}|_{0}^{x}$$
$$= 1 - e^{-\lambda x} \quad \blacksquare$$

It follows that, by the complement rule [5], $P(X > x) = 1 - \left(1 - e^{-\lambda x}\right)$. Thus,

$$P(X > x) = e^{-\lambda x} \tag{E.12}$$

Series and Parallel Systems: Basic Assumptions

Based upon the previous definitions, other assumptions need to be stated. First, all the n system sub-component service life, i.e., X, is a random variable exponentially distributed during the observed service life, t.

$$F_X(t) = P\{X \leq t\} \tag{E.13}$$

Since X is exponential distributed, and *t*, cannot take negative values; from Eqs. E.10 and E.13, the cumulative distribution function can be written as

$$F_X(t) = 1 - e^{-\lambda t}, \tag{E.14}$$

where λ is the failure rate in $1/t$ unit time (i.e., $1/h$). Consequently, from Eq. E.9, the probability function can be defined as

$$f_X(t) = \lambda e^{-\lambda t}. \tag{E.15}$$

Also, for every n system sub-component, $1 \leq i \leq n$, failure rate, FR, is constant. That is,

$$\lambda_i(t) = \lambda_i \tag{E.16}$$

Since t represents an interval of time, $[0, t]$, where a failure can occur, the reliability of any i^{th} system sub-component, $1 \leq i \leq n$, gives rise to

$$R_{X_i}(t) = P\{X_i > t\} = e^{-\lambda_i t}, \tag{E.17}$$

which in turn implies that

$$\lambda_i = -\frac{\ln (R_{X_i}(t))}{t} \tag{E.18}$$

Since all n system sub-component lives are exponentially distributed. That is, sub-component FR is time independent, we have

$$P\{X_1 \wedge X_2 \wedge \ldots \wedge X_n > t\} = P\{X_1 > t\} \times P\{X_2 > t\} \times \ldots \times P\{X_n > t\} \tag{E.19}$$

Subsequently, from Eqs. E.17 and E.19, gives rise to the definition of the failure rate of the system, λ_s

$$\begin{aligned} R_{X_s}(t) &= R_{X_1}(t) \times R_{X_2}(t) \times \ldots \times R_{X_{n-1}}(t) \times R_{X_n}(t) \\ &= e^{-\lambda_1 t} \times e^{-\lambda_2 t} \times \ldots \times e^{-\lambda_{n-1} t} \times e^{-\lambda_n t} \\ &= e^{-t \sum_{i=1}^{n} \lambda_i} \\ &= e^{-\lambda_s t} \\ &\Rightarrow \lambda_s = \sum_{i=1}^{n} \lambda_i \end{aligned} \tag{E.20}$$

which in turn implies the following

$$R_{X_s}(t) \quad = e^{-\lambda_s t}$$

$$\Rightarrow \lambda_s = -\frac{\ln\left(R_{X_s}(t)\right)}{t} \tag{E.21}$$

Consequently, from Eqs. E.20 and E.21, the definition of mean time between failures[29], MTBF, can be derived as follows

$$MTBT = \mu = \frac{1}{\lambda_s}$$

$$\mu = \frac{1}{\sum_{i=1}^{n} \lambda_i} \tag{E.22}$$

Or

$$\mu = \frac{1}{-\frac{\ln\left(R_{X_s}(t)\right)}{t}} \tag{E.23}$$

Reliability of Series Systems

Graphically, a series system can be seen as a sequential arrangement of components which are simply placed one after another (see Figure E.1). A series system is a configuration such that, if any one of the system sub-components fails, the entire system fails. Conceptually, a series system is one that is as weak as its weakest link.

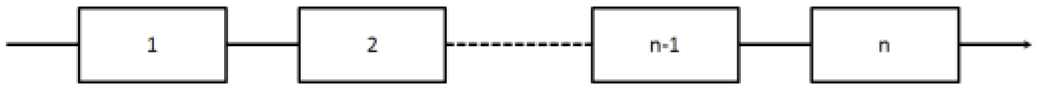


Figure E.1. Representation of a Series Systems of "n" components.

Mathematically, the reliability of a series system, i.e., Lusser's Law, is defined as follows: "*The reliability of a series system, i.e.,* R_{X_s}*, is equal to the product of the reliability*

[29] MTBF is a statistical mean value for error-free operation of a system sub-component.

of its component subsystems, ie., R_{X_i}*, where* $1 \leq i \leq n$*, if their failure modes are known to be statistically independent*." Eq. E.24 describes the Lusser's Law.

$$R_{X_s} = R_{X_1} \times R_{X_2} \times \cdots \times R_{X_{n-1}} \times R_{X_n} \tag{E.24}$$

A modern formulation of the series system reliability can be expressed as

$$R_{X_s}(t) = \prod_{i=1}^{n} R_{X_i}(t), \tag{E.25}$$

Example

Four subsystems are reliability-wise in series and make up a system. Subsystem 1 has a reliability of 98.5%, subsystem 2 has a reliability of 99.7%, subsystem 3 has a reliability of 96.3%, and subsystem 4 has a reliability of 98.2% for a mission of 100 hours. What is the overall reliability of the system for a 100-hour mission? What is the failure rate, FR, of the systems for a 100-hour mission?

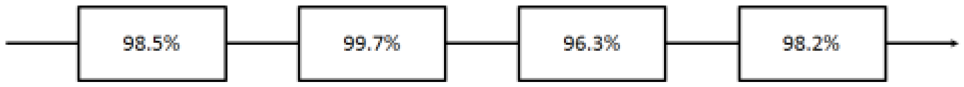


Figure E.2. Graphical representation for the given example.

Since the reliabilities of the subsystems are specified for 100 hours, the reliability of the system for a 100-hour mission is (Eq. E.25):

$$R_{X_s} = R_{X_1} \times R_{X_2} \times R_{X_3} \times R_{X_4}$$
$$R_{X_s} = 0.985 \times 0.997 \times 0.963 \times 0.982$$
$$R_{X_s} = 0.92868656697$$
$$R_{X_s} = 92.86\%$$

The FR of the system for a 100-hour mission is (Eq. E.21):

$$\begin{aligned}\lambda_s &= -\frac{\ln\,(R_{X_s}(t))}{t}\\ &= -\frac{\ln\,(R_{X_s}(100))}{100}\\ &= -\frac{0.9286}{100}\\ &= -\frac{-0.0740}{100}\\ &= 0.0000740\end{aligned}$$

Since the system FR is $\lambda_s = 0.0000740$, then the system MTBF (Eq. E.22) is

$$\begin{aligned}MTBT = \mu &= \frac{1}{\lambda_s}\\ \mu &= \frac{1}{\lambda_s}\\ \mu &= \frac{1}{0.0000740}\\ \mu &= \frac{1}{0.0000740}\\ \mu &= 14204.545\end{aligned}$$

Reliability of Parallel Systems

Graphically, a parallel system can be seen as an arrangement of components such that, s long as not all the system components fail, the entire system works (see Figure E.3).

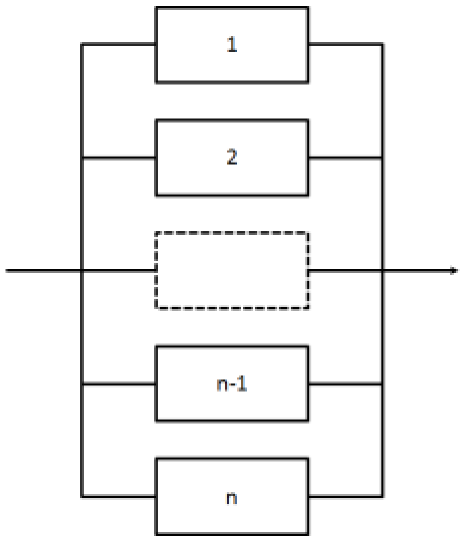


Figure E.3. Representation of a Parallel Systems of "n" components.

Mathematically, the reliability of a parallel system is defined in Eq. E.26.

$$R_{X_s} = 1 - (1 - R_{X_1}) \times (1 - R_{X_2}) \times \cdots \times (1 - R_{X_{n-1}}) \times (1 - R_{X_n}) \qquad \text{(E.26)}$$

Rearranging terms in Eq. E.26, another formulation of the parallel system reliability can be obtained

$$R_{X_s}(t) = 1 - \prod_{i=1}^{n} (1 - R_{X_i}(t)) \qquad \text{(E.27)}$$

However, behind Eq. E.27 lies a whole body of probabilistic knowledge. To illustrate, we analyze a simple parallel system composed of two sub-components. The system can survive observed service life, t, if and only if the first component , or the second component or both survive t (see Figure E.4). First recall from probability theory that,

$$P(A \cup B) \cup P(\bar{A} \cap \bar{B}) = 1. \qquad \text{(E.28)}$$

From Eqs. E.17, E.19, E.20, and E.28, we can formally define the reliability of a parallel system can determined as follows. The reliability of a parallel system can be expressed as:

$$R_{X_s}(t) = P\{X_s > t\} \tag{E.29}$$

where, $P\{X_s > t\}$ represents the probability of two independent events that occur in sequence. Therefore, Eq. E.29 can be reformulated as follows

$$R_{X_s}(t) = P\{X_1 > t \lor X_2 > t \lor (X_1 > t \land X_2 > t)\} \tag{E.30}$$

From Eq. E.28, it finally implies that

$$P\{X_1 > t \lor X_2 > t \lor (X_1 > t \land X_2 > t)\} = 1 - (1 - X_1 > t) \times (1 - X_2 > t) \tag{E.31}$$

Subsequently, Eq. E.31 can be stated as by replacing terms (see Eq. E.17)

$$P\{X_1 > t \lor X_2 > t \lor (X_1 > t \land X_2 > t)\} = 1 - (X_1 < t) \times (X_2 < t) \tag{E.32}$$

$$P\{X_1 > t \lor X_2 > t \lor (X_1 > t \land X_2 > t)\} = 1 - \left(1 - e^{-\lambda_1 t}\right) \times \left(1 - e^{-\lambda_2 t}\right) \tag{E.33}$$

resulting in

$$R_{X_s}(t) = 1 - \left(1 - R_{X_1}(t)\right) \times \left(1 - R_{X_2}(t)\right) \tag{E.34}$$

Consequently, this approach can be easily extended to n number of parallel system sub-components.

$$R_{X_s}(t) = 1 - \left(1 - R_{X_1}(t)\right) \times \left(1 - R_{X_2}(t)\right) \ldots \times \left(1 - R_{X_{n-1}}(t)\right)\left(1 - R_{X_n}(t)\right) \qquad \text{(E.35)}$$

Using instead, the probabilistic formulation of $R_{X_s}(t)$, Eq. E.34, we can obtain system MTBF (μ) for an arbitrary observed service life, t. For the hypothetical example:

$$R_{X_s}(t) = 1 - \left(1 - e^{-\lambda_1 t}\right) \times \left(1 - e^{-\lambda_2 t}\right) \qquad \text{(E.36)}$$

$$= e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t} \qquad \text{(E.37)}$$

$$\Rightarrow MTBF = \mu = \int_0^\infty R_{X_s}(t)dt = \int_0^\infty \left(e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}\right) dt \qquad \text{(E.38)}$$

$$\mu = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2}$$

Finally, one can calculate system FR, λ_{X_s}, from Eqs. E.15 and E.25 as indicated

$$\lambda_{X_s}(t) = \frac{f_{X_s}(t)}{R_{X_s}(t)}$$

resulting in

$$\lambda_{X_s}(t) = \frac{\lambda_1 e^{-\lambda_1 t} + \lambda_2 e^{-\lambda_2 t} - \lambda_1 \lambda_2 e^{-\lambda_1 \lambda_2 t}}{e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}}$$

Example

Let a parallel system be composed of two sub-components, each with a $FR = 0.01$ and observed service life $t = 10$ hours, only one is needed for systems success. Then, total system reliability, by both calculations, is:

$$R_{X_1}(t) = P\{X_1 > 10\}$$

$$= e^{-\lambda_1 t}$$

$$= e^{-(0.01)10}$$

$$= 0.9048$$

$$R_{X_2}(t) = P\{X_2 > 10\}$$

$$= e^{-\lambda_2 t}$$

$$= e^{-(0.01)10}$$

$$= 0.9048$$

$$R_{X_s}(10) = 1 - \left(1 - R_{X_1}(10)\right) \times \left(1 - R_{X_2}(10)\right)$$

$$R_{X_s}(10) = 1 - (1 - 0.9048) \times (1 - 0.9048)$$

$$R_{X_s}(10) = 1 - (0.0952) \times (0.0952)$$

$$R_{X_s}(10) = 0.9909$$

$$R_{X_s}(10) = e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1 + \lambda_2)t}$$

Because $\lambda_1 = \lambda_2$

$$R_{X_s}(10) = 2e^{-\lambda t} - e^{-(2\lambda)t}$$

$$R_{X_s}(10) = 2e^{-(0.01)10} - e^{-20(0.01)}$$

$$R_{X_s}(10) = 2e^{-(0.1)} - e^{-(0.2)}$$

$$R_{X_s}(10) = 0.9909$$

Mean time between failures in hours:

$$\mu = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2}$$

$$\mu = \frac{1}{0.01} + \frac{1}{0.01} - \frac{1}{0.02}$$

$$\mu = \frac{2}{0.01} - \frac{1}{0.02}$$

$$\mu = 150$$

References

1 Bajenescu, T.-M.I., Bazu, M.I.: Component Reliability for Electronic Systems. Artech House Publishers (2009)

2 Savage, L.J.: The foundations of statistics. Wiley & Sons (1954)

3 Storey, N.: Safety-Critical Computer Systems. Addison-Wesley Longman (1996)

4 Taboga, M.: Lectures on probability theory and mathematical statistics (2nd Edition). CreateSpace Independent Publishing Platform (2012)

5 Villemeur, A.: Reliability, Availability, Maitainability and Safety Assesment: Volume 1 - Methods and Techniques. Wiley & Sons (1992)

6 Yates, D.S., Moore, D.S., Starnes, D.S.:The Practice of Statistics (2nd ed.). New York: Freeman. (2003)