Evolutionary multi-objective optimization of QoS-Aware Publish/Subscribe Middleware in electronics production

Ivan M. Delamer, Jose L. Martinez Lastra*

Institute of Production Engineering, Tampere University of Technology, P.O. Box 589, FIN-33101 Tampere, Finland

Received 22 April 2005; received in revised form 17 October 2005; accepted 21 December 2005
Available online 6 March 2006

Abstract

Computer-Aided Manufacturing using XML (CAMX) production systems are built on Message-Oriented Middleware Frameworks, offering standards-based communication among machines and control software applications. CAMX Frameworks implement Publish/Subscribe of XML messages through an entity called the Message Broker (MSB), which provides the messaging service using a Simple Object Access Protocol (SOAP) interface. In order to create scalable frameworks, distributed MSB systems are deployed. However, the topology optimization problem arises, as clients need to be assigned to one of many MSB nodes. The problem is strictly NP-hard, and multiple optimization criteria are conflicting. A solution considering real-time systems was developed based on Evolutionary Multi-Objective Optimization techniques. The Framework Optimization Algorithm (FOA) is designed to work with various topologies, including federated frameworks, locally distributed clusters, and mixed environments including embedded middleware nodes. The developed FOA was tested for a case scenario based on a flexible manufacturing system running on a distributed CAMX framework, and proved to robustly converge to the optimal topology. Convergence was achieved within few seconds, demonstrating the suitability of FOA for rapid topology reconfiguration in response to changes in the system.

Keywords: Evolutionary algorithms; Rapidly reconfigurable manufacturing systems; Event-based systems; Electronic equipment manufacture; Manufacturing automation software

1. Introduction

Current trends for manufacturing systems indicate increasingly intensive and complex information exchange at the factory floor, together with ever more demanding requirements for flexibility and reconfigurability. Middleware has long been proposed as enabling technology to seamlessly integrate heterogeneous components, to the point that middleware is no longer an option but a critical component of any distributed system. Within the domain of middleware, which is a $10 billion industry on itself, Message-oriented Middleware (MOM) provides a scalable and real-time solution for asynchronous, event-based systems, such as manufacturing systems built of heterogeneous machines, devices, and software applications. The significance of MOM has been underlined by recent European research roadmaps (Bouyssounouse and Sifakis, 2005; ITEA, 2004), having stated that MOM is a candidate technology for enabling real-time support in distributed environments, where other middleware solutions such as CORBA, JINI and .NET struggle with interoperability and dynamic reconfigurability.

The Electronics Production Industry, pushed by fierce needs for cost reduction and mass customization, has been one of the early adopters of this technology in the form of Computer-Aided Manufacturing using XML (CAMX) open standards (Dugenske et al., 2000). CAMX systems are built on a standardized Framework, which provides a Web-based messaging service to communicating machines using Publish/Subscribe for event distribution through a Message Broker (MSB) system. The messages exchanged through the MSB have been also standardized as XML messages, with well-defined syntax and semantics.
While MSB systems were originally implemented as a single server providing XML Publish/Subscribe as a Web Service, recent advances have transformed CAMX Frameworks into distributed MSB systems (Delamer et al., 2005a, b). These distributed systems may take the form of locally distributed MSB clusters, MSB federations, Peer-to-Peer overlay networks with embedded MSB nodes, or a combination of (all) the preceding. For any of the aforementioned topologies, any given MSB node services a set of machines that publish and/or subscribe to messages associated with manufacturing process events. Every machine is serviced by one and only one MSB node, providing architectural transparency of the underlying infrastructure.

A critical task that must be undertaken by the distributed middleware framework is that of transparent topology management and optimization (Delamer et al., 2005b). Inappropriate or lack of topology optimization can degrade productivity due to poor real-time performance, increased network use, inefficient load balancing between nodes, and CPU overload of embedded control devices with limited computational resources. The problem to solve is therefore to dynamically assign equipment, devices, and control software applications to one of several available MSB nodes, in a way that optimization goals such as Quality of Service (QoS), network utilization and CPU utilization are simultaneously considered.

CAMX messages are typically related to events in a manufacturing process, and therefore are of interest to other machines, devices, and control software applications that share the same process. In this way, machines that share a given manufacturing process offer a natural grouping that could be exploited to optimize the topology of the underlying messaging infrastructure. However, these groups are not disjoint, due to machines participating in more than one process, and due to software applications that are not process-specific and thus communicate with machines participating in different processes, making the optimization problem non-trivial. The problem complexity is NP-hard as there is no simple heuristic to determine if a given solution is the optimal one. Moreover, the problem is aggravated by conflicting optimization criteria, such as QoS vs. load balancing, which suggest divergent solutions.

Considering the characteristics of the problem, a combinatorial optimization meta-heuristic was sought. An evolutionary strategy was chosen as this type of algorithm incorporates learning behavior for good partial solutions, which in the case at hand accommodates well to consolidated groups of machines that share given processes. The genotypes that encode topology solutions, the genetic operators, and the FFs were specialized to the particular problem domain.

The rest of this paper is structured as follows. Section 2 reviews the CAMX approach for communications in the electronics production industry. Section 3 presents an architectural approach for creating distributed Publish/Subscribe middleware that is tailored for the CAMX case. Section 4 considers the topology optimization problem, which is solved by using an evolutionary multi-objective algorithm. Section 5 presents tests and results for a representative recreated scenario of a flexible manufacturing system and analyses the outcome. Section 6 summarizes the achievements and presents some concluding remarks.

2. Computer-Aided Manufacturing using XML review

CAMX refers to a set of standards that specify Web-based communication protocols for the electronics production industry. The standardization activities for communications and data exchange interfaces were initiated by iNEMI (formerly the National Electronics Manufacturing Initiative), an industry-led consortium of over 60 large electronics manufacturers and other organizations, and carried out by an ad hoc conglomerate of leading research institutions and industrial organizations. CAMX comprises the IPC-25xx family of standards, which are endorsed by IPC, Association Connecting Electronics Industries, an ANSI accredited standards body with membership of over 2000 companies and 1000 individuals.

2.1. CAMX history

In 1996, the iNEMI roadmapping activity (NEMI, 1996) revealed that factory information and communication systems were too expensive, hardly interoperable, required huge integration efforts, and did not provide enough visibility. Estimates showed that integrating equipment and factory information systems cost as much as 4–20 x the cost of the software itself. iNEMI subsequently launched the Plug&Play Factory project, with the goal to design, develop and demonstrate a technical infrastructure to enable the deployment and replacement of both electronics assembly equipment and electronics assembly software in a fraction of the time and at a fraction of the cost normally associated with those activities. The resulting infrastructure is based on asynchronous XML message passing between equipment and software applications, accommodating for the required data complexity, and multi-vendor and multi-platform systems existing in factories (Dugenske et al., 2000).

The work initiated in the Plug&Play Factory project was continued by several follow-up projects starting in 1999, which targeted at standardizing the syntax, content and semantics of XML messages. These projects were mostly industry-led, but coordinated by research organizations in Europe and USA, and are an ongoing effort to accommodate the most current requirements. The CAMX concept was initially demonstrated by pilots at Georgia Tech., USA, and Tampere University of Technology, Finland (Siltala et al., 2003). Followed by several pilots in industrial environments, CAMX is being used in production in several factories since 2004. During that year, Flexlink Automation Oy has deployed four assembly lines with 40+ interconnected CAMX machines and
software applications (Martinez Lastra et al., 2005), and
the NACOM Corporation has reported savings of
$400,000 for a $130,000 investment in CAMX (NACOM,
2004).

2.2. CAMX framework

CAMX defines an event-based conversational frame-
work based on exchange of standardized XML messages
through Ethernet networks. In order to share a common
interface that can be supported by a wide range of
manufacturing equipment and that is vendor- and plat-
form-independent, the IPC-2501 standard defines the
middleware specification used to exchange CAMX data
(IPC, 2003). TheIPC-2501 standard specifies a MOM
approach utilizing as central element, the MSB, for
distribution of messages. The MSB supports both Pub-
lish/Subscribe messaging for distribution of manufacturing
process events, and Point-to-Point messaging for manu-
factoring control commands, request/responses, and ser-
vice invocations (Fig. 1).

The standard syntax, content and semantics of CAMX
messages is defined in the IPC-25xx series of standards. In
particular, the IPC-254x series of standards define mes-
sages associated manufacturing events in shop-floor
equipment. The IPC-255x series in turn defines messages
for bi-directional communication between equipment and
software applications, mostly used for supervisory control
functions. Generic messages exist that are applicable to all
equipment in the domain, as well as specific messages for
equipment such as pick-and-place machines, screen prin-
ters, reflow ovens, final assembly equipment, and so on.

CAMX also considers other domains than manufacturing.
Computer-Aided Design (CAD) of electronics pro-
ducts is facilitated by the IPC-251x standards and the IPC-
2581 standard. The Computer-Aided Manufacturing
(CAM) domain is considered by the IPC-2531 standard,
that specifies the file format for process recipes. Finally,
the supply chain aspects are considered in the IPC-257x series
of standards, which leverage the work done by RosettaNet.

2.3. Quality of Service model

One of the implementation challenges for CAMX
systems is to provide adequate QoS differentiation for
data traffic, in order to improve real-time performance
(Delamer et al., 2004a, b). CAMX is aimed at process-level
control, where events and associated messages are stochas-
tic in nature, as opposed to deterministic hard-real time
of periodic device-level control. As a consequence, a soft
real-time model for QoS needs to be employed.

The QoS issue has been previously considered by the
authors, resulting in a soft real-time QoS model (Delamer
et al., 2004c; Delamer and Martinez Lastra, 2006). The
model considers four differentiated flows, where XML
messages are assigned according to their purpose in the
manufacturing systems. The four categories are messages
for closed-loop real-time control, for supervisory control,
for operator control, and for other purposes (default). The
flows are differentiated using strict priority scheduling,
being closed-loop control the highest priority flow.

3. Distributed Publish/Subscribe frameworks

One of the challenges for implementing dependable
MSB-based Frameworks is that they appropriately scale to
handle large, information-intensive manufacturing systems
with associated large volumes of message exchange
(Delamer et al., 2004a). Software or hardware scale up by
performance optimization is limited by the complexity of
the MSB processes. In turn, a scale out approach in which
the MSB system is distributed to multiple processing nodes
needs to be applied. Several techniques and architectures
for parallel and distributed computing have been developed
for solving similar problems in the domain of Enterprise
Information Systems and Web Servers (Delamer et al.,
2004b). Applicable ones have been previously adapted for
the CAMX case by the authors (Delamer et al., 2005a, b),
and are briefly reviewed in the following subsections.

3.1. Federated frameworks

One scale out approach proposes a federated infrastruc-
ture, with MSB nodes servicing localized domains and
providing transparent interconnection between domains
(Delamer et al., 2005a). This approach is especially well
suited for factories running several distinct processes with a
certain degree of independence, e.g. several independent
assembly lines. The majority of CAMX messages are
typically exchanged within equipment and software appli-
cations belonging to the same process. A local MSB node
will efficiently service the messaging requirements of the
equipment and applications for a given process, and will
transparently exchange through other federated MSB nodes the smaller volume of messages destined for other localities. Servicing the majority of messaging volume locally eliminates large amounts of converging traffic at a centralized node, which can potentially incur in delays or increased network infrastructure costs, and also simplifies administration functions, whether performed by operators or autonomously.

Making use of this architecture, frameworks can easily scale in order to handle increased volumes by adding more nodes to the federation. The interactions between equipment and software applications with a local MSB node maintain the standard IPC-2501 service profile, so the federation is viewed from the client perspective as a single entity. This type of architecture transparency ensures that no changes are needed on the client-side of the interface, whether interacting with a single-server MSB or a federated framework.

3.2. Clustered frameworks

An alternative scalable approach to federated frameworks is to maintain a centralized approach by creating locally distributed MSB clusters. This approach is extensively used in order to improve the performance of Web Servers, which are increasingly stressed due to the increase of number of clients and access hits, and due to the increasing complexity of content and services offered. The goal of locally distributed Web servers, or Web server farms, is to evenly distribute the load among multiple servers, but still provide clients a single virtual interface. Several proficient architectures for locally distributed Web servers have been surveyed by Cardellini et al. (2002), who have also defined a taxonomy for this type of systems.

Given that CAMX MSBs can be classified as Web Servers that provide a dynamic Web Service, locally distributed MSB clusters can be implemented in the same way as Web Servers using replicated application servers and a large back-end database. However, given the very dynamic nature of a messaging system, where data unique to each message and is therefore always changing, replication through multiple servers creates a huge overhead and degrading real-time performance. This overhead can be avoided if a single queue per client is kept at a single server in the cluster, and therefore all requests for a given client are directed to the same server (Delamer et al., 2005b). This semi-static assignment also eliminates costly routing hardware, which would otherwise be needed at the front-end of the cluster to route every message to an appropriate server.

3.3. Unified architecture

The two previous approaches can be considered as complementary: a locally distributed cluster can be used within a federation to increase the capacity of a federated node. However, the proposed architecture for locally distributed, cluster-based CAMX frameworks bears significant resemblance to federated frameworks, given that a clustered Broker element services a relatively static set of clients, similar to a federated Broker. A unified architecture that uses the same type of Broker elements, and that is applicable to both federating MSBs and to creating locally distributed clusters, can be achieved (Delamer et al., 2005b).

The advantages for having a unified architecture lie on both design and implementation simplification, and improved flexibility and agility at run time. Having a single architecture simplifies the development of systems because a reduced set of element types needs to be developed and integrated. At run time, the ease of scaling up the framework is enhanced by deploying the same type of elements for adding messaging resources as needed, whether dynamically adding localities to a federation, or dynamically adding to (or forming) a locally distributed cluster.

Given that the same type of broker elements are used for federations and local clusters, the decision of whether to include a new broker node as a federated node or as part of a locally distributed system can be made based on the concept of logical and physical proximity. Elements that are contained within a same Local Area Network (LAN) are in physical proximity, and are eligible to form a locally distributed cluster. Likewise, elements that share a manufacturing process and therefore exchange the majority of information among themselves are in logical proximity, and are therefore eligible to be serviced within a same locality, possibly forming a locally distributed cluster. Elements that are at physical or logical distance are candidates to form a federated locality.

3.4. Embedded CAMX middleware

As CAMX Frameworks become increasingly distributed, the possibility is given for embedding MSB nodes in equipment with sufficient computational resources rather than deploying on a dedicated server. If sufficient MSB nodes can be embedded, the need for introducing supplementary Web Servers to a manufacturing system in order to provide the CAMX Framework is eliminated. Nevertheless, incorporating embedded MSB nodes increases the granularity and the complexity of the topology, making automatic topology optimization ever more necessary.

However, embedding the MSB business logic might become too expensive in computational resources for some embedded systems. Taking into account the current state of the art in embedded computing, embedded MSB nodes should only be considered for implementation in high-end equipment with powerful embedded CPUs and in high-end computers running CAMX software applications.

An embedded MSB node in a distributed CAMX Framework can be considered as any other MSB node deployed on a dedicated server. If located in proximity to other nodes, it can form a locally distributed cluster.
Otherwise, it can become a federated node. A mixed environment contemplating the different possible types of MSB nodes and the resulting CAMX infrastructure is illustrated in Fig. 2.

4. Optimization of distributed Publish/Subscribe CAMX middleware

When working with distributed CAMX frameworks, there will always be multiple solutions to the problem of assigning clients to one of many MSB nodes. Every solution will have different performance characteristics, given that every MSB node has different capacity, that every client produces a different amount and type of events, and that different topologies require different volumes of message traffic between MSB nodes. Worst-case behavior of a distributed system with inappropriate topology far exceeds that of a centralized system. A need for topology optimization naturally arises.

In event-based systems such as CAMX manufacturing systems, where processes that generate events have a significant stochastic component and rapid reconfiguration must be supported, topology optimization at design time is insufficient. Therefore, a more robust method that is able to operate at run time and rapidly adjust to changes in the system needs to be employed. The next subsections will discuss the different optimization goals and scenarios where optimization is required, and will introduce an optimization algorithm based on multi-objective evolutionary techniques.

4.1. Optimization goals and scenarios

When optimizing distributed CAMX framework, several optimization goals arise. These can be summarized as follows:

- To balance the processing load among the various MSB nodes.
- To reduce the overall processing load of the framework.
- To reduce the amount of interactions between MSB nodes, which arise when a message must be routed from publisher to subscriber through more than one intermediate node.
- To improve the QoS by favoring the transfer of time-critical messages.

If considered individually, the optimization goals would lead to different and incompatible type of solutions. Given the contradictory nature of the different criteria, trade-off solutions are needed.

CAMX messages are typically related to events associated to a manufacturing process, and therefore are of interest to other machines, devices, and control software applications that share that same process. As a consequence, the majority of messages are typically exchanged within restricted sets of clients who share a process. This motivating characteristic, that messages related to a given process are bounded within a limited domain, suggests that the optimization goals could be achieved by grouping related clients so that they are serviced by the same MSB node.

However, the groups of related clients are not disjoint due to machines participating in more than one process, and due to software applications that are not process-specific and communicate with machines participating in different processes. This overlapping of the groups of related clients cannot be resolved in any simple way. Also, some of these groups are too big to be serviced by a single MSB node and have to be broken down. The problem complexity is NP-hard as there is no simple heuristic to determine if a given topology solution is the optimal one: every feasible solution must be compared.

The need for topology optimization arises in a variety of scenarios. When working with a federated type of messaging framework, the design problem of how to create localities given a set of equipment and a number of MSB nodes arises. This design problem may occur when creating a new manufacturing system, or when an existing
manufacturing system grows and the messaging infrastructure needs to be federated to accommodate increasingly demanding conditions.

When working with a cluster-based messaging framework, the optimization problem of how to assign clients to a given server in the cluster arises. Once again, by grouping related clients, the interactions between servers in the cluster are reduced, reducing the load on the network as well as the processing overhead.

In scenarios where some clients have spare computational resources, such as application servers holding a MES system or a high-end machine equipped with a high capacity computer, MSB nodes can be embedded. In this case, equipment providing message broking capabilities will service a reduced locality formed by other equipment sharing the same manufacturing process. When working with mixed environments of local clusters, federated nodes and embedded MSB nodes, the increased complexity requires an automated mechanism to dynamically and autonomously adapt the topology in real time.

4.2. Framework Optimization Algorithm

The problem of optimizing the topology of a distributed CAMX framework is strictly NP-hard in complexity, given that the number of solutions rises exponentially with the number of clients and the number of MSB nodes. This type of complexity makes the problem unfeasible to be solved using deterministic algorithms. In addition, the optimization goals were mentioned to be conflicting, e.g. reducing inter-MSB interactions tends to group clients, while balancing the MSB load tends to distribute them. Evolutionary algorithms have been extensively used to find good solutions to NP-hard problems at reasonable computational expense for similar types of problems, e.g. the bin packing problem (Brown and Sumichrast, 2005), or switched network topology optimization (Youssef et al., 2002). Moreover, Evolutionary Multi-Objective Optimization (EMOO) Algorithms have also shown many benefits and robustness for finding Pareto-optimal solutions to problems where the different optimization functions are in conflict with each other (Coello Coello, 2000).

While several canonical solutions have been created for some genetic and EMOO algorithms since the pioneering work of Holland (1975), these are not immediately suited for the CAMX Framework optimization problem. The limitations of canonic implementations are in both the type of evaluation of the fitness of solutions, which measure how good a particular topology is, and in the type of genetic operators. The particular complexity of the fitness evaluation of this case is due to the need to evaluate not only to which MSB each client is assigned, but also where these clients are located in relation to other clients. An ad-hoc EMOO algorithm was therefore produced, and is denominated Framework Optimization Algorithm (FOA).

FOA works with an initial parent population of solutions that is randomly generated. This random population introduces diversity in the initial seeding of the algorithm in order to facilitate the exploration of the entire solution space. For every evolution cycle, a child population is created from the parent population by mutating individual parents, or by combining two parent solutions. Subsequently, a set of fitness functions (FFs) associated to different optimization goals are evaluated for every parent and child solution. Utilizing different criteria, the fittest solutions are selected to form the new parent population that will be used in a successive iteration of the FOA.

4.2.1. Topology solution encoding

The solutions of FOA are encoded as a chromosome represented as an array of genes, where each gene corresponds to a particular CAMX client. A gene contains the information of which MSB is assigned to the particular client. This information is represented as an integer value, with each MSB node having assigned a unique identifying number. More formally, a chromosome $c_k$ can be specified as follows:

$$
\tilde{c}_k = (g_1 \; g_2 \; \cdots \; g_N), \quad k \in [1, P],
$$

$$
g_i \in [1, M], \quad i \in [1, N],
$$

where $M$ is the number of MSB nodes, $N$ the number of clients, $P$ the number of parent chromosomes.

4.2.2. Mutation and crossover operators

The mutation and crossover operations enable the FOA to evolve and find new, potentially optimized chromosomes. The mutation operator is intended to dynamically introduce new genetic material and diversify the population of chromosomes. The crossover (recombination) operator is intended to find improved solutions using existing genetic material.

Traditionally, mutation operators act by randomly inducing changes in gene values with a given probability. For the FOA type of chromosomes, the mutation operator acts by making a copy of a parent chromosome, which will become the child chromosome. Next, a random process is evaluated for each gene in the chromosome, and based on the result of each evaluation it is determined whether a mutation will occur or not. For those genes to be mutated, a valid integer number representing a MSB node in the framework is randomly selected, and the gene values in the child chromosome are replaced. The mutation operator is illustrated in Fig. 3(a).

Traditionally, crossover operators act by taking two parent chromosomes and combining their genetic structure to create a child chromosome, known as recombination process. The challenge for recombination relies on carrying over chromosome portions that have some significance, rather than carrying over randomly picked genes that may not create fit chromosomes when recombined with other randomly selected genes. Many evolutionary algorithms suffer from breaking good solutions during recombination.
For the current case, it has been noted that clients will form groups associated to MSB nodes. On occasions, individual groups will be good partial solutions. The design strategy for this operator is to be able to crossover and recombine at the group level, instead of at the gene level, in order to maintain potentially good partial solutions.

For the FOA type of chromosomes, recombination is achieved by selecting two chromosomes to act as parents. Next, one gene is randomly selected from one of the two parents, hereafter called the father. After this gene has been selected, all other genes in the father chromosome with the same value are also selected, and copied to the child chromosome. The remaining genes are filled with the value from the remaining parent, the mother chromosome. Given that fit solutions will be those that group related CAMX equipment, this strategy ensures that a group of potentially related clients is carried over from the father chromosome, and recombined with the existing groups in the mother chromosome. The crossover operator is illustrated in Fig. 3(b).

### 4.2.3. Fitness functions

Several FFs have been created that are used to evaluate how “good” or “fit for survival” a chromosome is regarding the different optimization criteria described in Section 4.1. In order to evaluate meaningful FFs, several parameters that describe a chromosome with sufficient detail need to be calculated. These parameters quantify the computational requirements for the different processes an MSB node performs. These processes are: receiving published events; delivering events to subscribers; and, delivering events from MSB to MSB if intermediate nodes are traversed.

In order to calculate these parameters, the set of published messages is considered for each client (gene) represented in the chromosome, with each message having an associated frequency. These values are used to accumulate the computational resources that are taken in the MSB node servicing the publisher, in the MSB nodes servicing each one of the subscribers, and in any intermediate nodes. Moreover, each subscriber can impose a particular QoS requirement, represented as a priority level (Delamer and Martinez Lastra, 2006), so that the accumulation is performed for each QoS class.

In order to quantify the computational resources required by different solutions, the measurement unit used by the FOA is the XML message parsing operation. Previous studies have shown that XML parsing dominates CPU usage in MSB processes, while routing and scheduling logic represent a minor component (Delamer et al., 2004c). Therefore, MSB capacity will be measured in parsing operations per second, and the loading of MSB nodes will be calculated according to the parsing operations per second that clients generate. Given that different types of interactions with the MSB framework generate different number of parsing operations, different numbers of operations are appropriately accumulated for client-to-broker interactions (event publishing), broker-to-broker interactions (inter-node delivery), and broker-to-client interactions (subscription delivery).

The use of multiple parameters to measure computational resource utilization enables future use of more precise units than XML parsing, if the need were to arise. Given that XML parsers and serialization techniques are continually being improved (Geer, 2005), and more complex routing algorithms are being developed for distributed environments, the XML parsing unit by itself might become inaccurate to describe resource utilization and might need to include business logic processing time. However, because accumulation is performed for each one of the three types of processes, new measurements that include both business logic processing time and XML parsing could be seamlessly integrated to the FOA.

Following the proposed approach, any given topology solution is described at the granularity level of every QoS priority for every MSB node. Each one of these values is calculated as the accumulated number of XML parsing operations per second. These values are directly proportional to the computational loading of an MSB node, and inversely proportional to the delays induced to messages belonging to the described class. These descriptive parameters are used as input to calculate the following FFs:

**FF1. Aggregate processing fitness:** This function evaluates the total number of parsing operations for all MSB nodes. The optimization goal is to reduce the total processing requirements of the distributed framework.

**FF2. Broker-to-broker processing fitness:** This function evaluates the total number of parsing operations that are due to broker-to-broker interactions. The optimization goal is to reduce network use, to reduce the avoidable processing load on MSB nodes caused by
multi-hop message routing, and to reduce the delivery delays caused by additional hops.

FF3. QoS-aware fitness: This function evaluates the number of parsing operations for each QoS class that has been described in Section 2.3, and produces a weighted aggregate. The largest weight factor is given to time-critical (higher priority) classes. The optimization goal is to reduce the number of parsing operations for QoS-critical messages, and thus reduce overall delivery delays of real-time messages, which could degrade productivity.

FF4. Loading variance fitness: This function evaluates the statistical variance of the loading of the different MSB nodes. The loading is calculated as percentage use of the capacity of MSB nodes. The optimization goal is to evenly distribute the processing load throughout the distributed framework.

FF5. Loading peaks fitness: The variance function is a good measure of how well balanced a distributed system is, but is not sensitive enough to detect loading peaks where resource utilization approaches 100%, or to detect MSB overloading. Delivery delays of messages rise exponentially as loading increases (Delamer et al., 2004c), so loading peaks are to be avoided. In order to reach this optimization goal, this ad-hoc function evaluates the loading of each MSB node and returns values that render unfit any chromosome that has one or more utilization peaks.

FF6. Distance fitness: This function evaluates the distance of a chromosome to a reference chromosome, and gives a measure of how similar or different two chromosomes are. The distance measure is calculated as the number of genes with different value. The optimization goal is to find solutions that are close to a given solution. This goal is required when an already implemented topology is to be optimized and as few changes as possible are sought. In particular, this optimization goal is critical for frameworks where symmetric topologies with identical performance are possible, e.g. a cluster of two identical servers. If this FF is not considered, oscillations could occur if successive optimization runs allocate a client to alternating servers. These oscillations generate significant overhead and performance degradation.

FF7. Weighted sum fitness: This function evaluates a weighted sum of the previous six FFs. The optimization goal is to find solutions that are good considering all criteria. Given that each function has its own scale, the values used for the weighted sum are normalized using the average fitness values for the current population.

4.2.4. Selection strategies

After the FOA produces a population of children chromosome and evaluates the different FFs for both the parent and children population, the fittest chromosomes are selected. By selecting between a merged population of parents and children, it is guaranteed that no involution (the new generation is less optimized than the previous generation) will occur. This selection strategy can be classified as a \((\mu + \lambda)\)-ES using the notation described by Whitley (2001). Even though some evolutionary algorithms give a low probability of survival to chromosomes with poor fitness in order to maintain diversity and overcome local minima, the FOA achieves diversity by implementing a higher probability of mutation and by the use of selection through multiple FFs.

A key design aspect of the FOA is therefore how to perform chromosome selection given the multiple FFs. The selection problem becomes remarkably challenging because the different FFs often yield contradictory results. Functions FF1 and FF2 typically put forward chromosomes that reduce overall loading by centralizing in a few nodes, while functions FF4 and FF5 tend to favor chromosomes that distribute the load. In addition, function FF3 is dominated by a small portion of exchanged messages, those that are time-critical, and therefore may be unfit according to the other functions, which consider all messages equally. Moreover, the results of function FF6 have no correlation with those of the other functions, and might be in agreement or not.

Having the aforementioned challenge in mind, the FOA implements a selection strategy in which a fixed number of chromosomes are selected for each FF. The FF7 selection contributes with the largest number of surviving chromosomes, given that it determines which solutions are good as a compromise of all optimization goals, which is the ultimate goal of the FOA. Nevertheless, by carrying on some solutions that excel in one of the optimization dimensions, genetic configurations that are particularly fit according to some of the FF1-6 criteria can be recombined to form strong solutions according to more than one criterion. This approach was first proposed by Schaffer (1985) and was denominated Vector-Evaluated Genetic Algorithm (VEGA). Schaffer referred to the described phenomena as “speciation”, or the evolution of particular “species” within a population.

4.3. Comparison to other meta-heuristics and rejected approaches

The presented problem can be classified as a combinatorial optimization problem, for which several meta-heuristics besides evolutionary algorithms have been proposed; the most prominent have been surveyed by Blum and Roli (2003). A number of these meta-heuristics fall in the category of trajectory methods, including Simulated Annealing, Tabu Search, GRASP, Variable Neighborhood Search, Iterated Local Search, and Guided Local Search. The fundamental principle for these algorithms is to start from an initial solution and to repeatedly try to find better solutions that are close in the solution space, thus following the gradient towards the optimal solution. In order to overcome local minima, different random or probabilistic factors are implemented to
introduce diversity and explore the entire optimization space, which differentiate the named approaches. A criticism to the trajectory methods is that they do not incorporate any learning behavior from the experience in previously explored areas of the optimization space. This is critical for the problem at hand, as the hyper-surface of the optimization space for FOA has no clear gradient and is very irregular. Learning behavior, as incorporated by evolutionary algorithms, can assist in creating new improved solutions based on good partial solutions rather than following a gradient, improving efficiency.

A meta-heuristic of particular interest for the current problem is Ant Colony Optimization, as it also incorporates learning behavior. In this type of algorithm, a set of ant agents simultaneously explores the optimization space, leaving a trace of pheromone. Regions of the optimization space that contain several good solutions and are visited often acquire more pheromone, creating positive feedback and attracting the ants towards more optimal regions. This approach is especially well suited for performing replaced optimizations after small variations in the input, as the traces of pheromone are only gradually modified, and the optimization is fast for finding similar yet more optimized solutions. This characteristic is well suited to the problem at hand, where periodic optimization is required after changes in the manufacturing system occur, and has been implemented in the FOA using the distance FF (FF6). The ant colony approach however only performs well when there are continuous regions of “good” solutions, in order to accumulate pheromone in a bounded area and create the desired effect. For the problem at hand, these regions do not exist as the optimization hyper-surface is very irregular, and the ant colony approach is therefore not well suited.

Within the arena of EMOO meta-heuristics, a number of approaches have been suggested besides the chosen VEGA approach, and have been surveyed by Coello Coello (2000). Among these, lexicographic ordering considers the multiple objectives according to a ranking of importance. In the FOA, the importance of different functions has been considered by the weights in the compromise FF (FF7).

Another set of approaches takes its foundation in game theory, and attempt to find an intersection of the optimal solution fronts created by the different functions, or to find points of minimal distance between the fronts. These approaches proved not to be applicable to the FOA as the optimal fronts were not continuous enough and too divergent among different criteria.

A further set of approaches proposes the use of gender, in order to only recombine solutions that have been selected through different FFs. This feature was ruled out for the FOA as good solutions can also be found by recombining chromosomes of the same gender.

Finally, a set of approaches proposes Pareto ranking of the current population for each generation, in order to guide the selection process towards the Pareto optimal solution. This approach was rejected in favor of VEGA speciation, as the latter facilitated greater diversity of the population in order to overcome local minima.

5. Framework Optimization Algorithm tests and results

An implementation of the FOA was produced in order to investigate behavior under different scenarios and with different variations of parameters, as well as for testing its feasibility in real-time applications. The following subsections will discuss the implementation of the FOA, an example CAMX model of a flexible manufacturing system used for testing the FOA, and experimental results using different algorithm parameters, and will present an analysis of the results in the context of real-time optimization.

5.1. FOA implementation

The FOA was implemented using the Java programming language, given that existing implementations of CAMX frameworks have also been programmed in Java and therefore integration would be eased.

The input required by the FOA implementation is the set of MSB nodes and their associated capacities, and the set of clients with their associated published messages and frequencies, as well as their subscriptions and QoS requirements. This input is given in an extended version of the standard Domain Configuration, which is defined by the IPC-2501 standard. The Domain Configuration is an XML file that holds information on MSB nodes, clients, publishing capabilities, and subscription interests, which must be supported by all CAMX-compliant frameworks. The MSB processing capacities are given in a custom extension parameter for each MSB XML element, and the message publishing frequencies and subscription QoS requirements are given in a custom extension parameter for each message XML element. By using the extensibility features of XML, the extended Domain Configuration is backwards compatible.

The FOA contemplates the optimization of potentially mixed distributed frameworks, which may incorporate federated nodes, locally distributed clusters, and embedded MSB nodes. When working in environments with federated localities, a FOA will only optimize one locality at a time. Therefore, the subset of MSB nodes and clients to be optimized can be configured. Moreover, if embedded MSB nodes are present, the mapping on clients and associated MSB nodes can be configured. The FOA will then adjust calculations given that no broker-client interactions are needed for embedded MSB node-client pairs.

Several configuration options have been made available in order to tune FOA parameters during the evaluation stages. The available settings are summarized in Table 1.

A simple Graphic User Interface (GUI) was produced in order to simplify the configuration of parameters. In addition, the GUI provides visualization of the current population of chromosomes and the evaluated of FFs, as well as commands for manually controlling optimization...
The proposed example is a scaled-down version of a real system, and has been chosen to be illustrative and not exhaustively representative. Further experiments with larger system models have also been carried out, which have been omitted for clarity and brevity, and have offered similar results to the proposed example.

Table 1
FOA configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent population size</td>
<td>20 chromosomes</td>
</tr>
<tr>
<td>Children produced per generation</td>
<td>60 chromosomes</td>
</tr>
<tr>
<td>Ratio of children generated using the crossover operator</td>
<td>40%</td>
</tr>
<tr>
<td>Ratio of children generated using the mutation operator</td>
<td>60%</td>
</tr>
<tr>
<td>Gene mutation probability</td>
<td>FF1: 10%</td>
</tr>
<tr>
<td>Selected survivors per generation</td>
<td>FF2: 1 chromosome</td>
</tr>
<tr>
<td></td>
<td>FF3: 1 chromosome</td>
</tr>
<tr>
<td></td>
<td>FF4: 4 chromosomes</td>
</tr>
<tr>
<td></td>
<td>FF5: 3 chromosomes</td>
</tr>
<tr>
<td></td>
<td>FF6: 0 chromosomes</td>
</tr>
<tr>
<td></td>
<td>FF7: 10 chromosomes</td>
</tr>
<tr>
<td>Weight indices for weighted sum fitness function (FF7)</td>
<td>FF1: 0.1</td>
</tr>
<tr>
<td></td>
<td>FF2: 0.1</td>
</tr>
<tr>
<td></td>
<td>FF3: 1.0</td>
</tr>
<tr>
<td></td>
<td>FF4: 0.3</td>
</tr>
<tr>
<td></td>
<td>FF5: 0.7</td>
</tr>
<tr>
<td></td>
<td>FF6: 0</td>
</tr>
</tbody>
</table>

5.2. Test scenario

In order to test the FOA, a CAMX model of an example flexible manufacturing system was produced. The system is comprised of three assembly cells, each composed of a conveyor system for material handling, two robots, and a cell controller node. Additionally, the system has a line manager node, in charge of scheduling and process orchestration, and a line monitor node, which collects data and produces Overall Equipment Effectiveness (OEE) metrics for the machines. This scenario is illustrated in Fig. 51.

The Domain Configuration XML file that represents the system was produced, defining the messages that are published by the different machines and software applications, as well as the subscriptions. Altogether, 140 published messages were included, as well as 230 subscriptions. The QoS requirements of subscriptions were appended as an extension XML parameter, and were established by determining if the messages were used in a process control loop, in a supervisory control loop, or for other functions (Delamer and Martinez Lastra, 2006). The frequencies of each published message were also appended as an extension XML messages, and were determined by adapting observed values at the pilot CAMX assembly line.

5.3. Test results

A set of configurations of the FOA parameters was formulated in order to test the behavior variation under different settings. In terms of population size, two configurations were tested: 20 parents and 60 children per generation, and 40 parents and 100 children per generation. In terms of selection strategy, three configurations were tested: using only functions FF1 through FF6 in order to stimulate speciation, using only function FF7 to promote compromise solutions, and splitting surviving chromosomes half for FF1–FF6 and half for FF7. The weights assigned to the different parameters of FF7 (weighted sum fitness) gave the biggest selection influence to QoS fitness (FF3) and Loading Peaks fitness (FF5). The number of generations (algorithm iterations) was arbitrarily set to 1000.

Before running the FOA, the optimal topology was manually determined through a process of speculation and iteration. The optimal configuration was determined to be one in which conveyors and robots are assigned to the MSB node embedded in the corresponding cell controller machine, and the line manager and line monitor clients are assigned to the remaining MSB nodes. The goal of the experiments was thereafter to test if the FOA, under different configurations, converged robustly to the aforementioned configuration. In this case, “robustly” was quantitatively assessed by resetting and executing the FOA 50 times for each case, and verifying if convergence was consistent. The resulting state of the FOA for a particular optimization run that converged to the Pareto-optimum solution is illustrated in Fig. 6.

The FOA configurations and outcome of the tests are summarized in Table 2. The use of pure speciation-based selection, which optimizes by finding and recombining solutions that are optimal in only one dimension, only sporadically converged to the optimal solution. The FOA in this configuration converged to solutions that were typically good but sub-optimal, failing to overcome local minima in the multi-dimensional optimization space. Likewise, the use of selection based on compromise solutions converged to sub-optimal solutions that are local minima. This situation is illustrated in Fig. 7, which depicts a simplified two-dimensional optimization space and an arbitrary function exposing several local minima. Once
a population of chromosomes is dominated by solutions around a particular local minimum, the crossover and mutation operators are unable to create new chromosomes that are distant enough to find other minima. However, a combined strategy proved to maintain enough diversity of chromosomes, and the FOA consistently converged to the Pareto-optimal solution for both population size settings.

Having determined that a hybrid selection strategy produced the most consistent convergence to the optimal solution, several experiments followed in order to test how sensitive convergence was to parameter values. The following values were tested in several combinations:

- The mutation probability was set to 5%, 10%, 15% and 20%.
- The mutation/crossover generation ratio was set to 30/70, 40/60, 50/50, 60/40 and 70/30.
- The selection cardinality for each FF was varied, setting the weighted FF (FF7) to 25%, 50% and 75% of selections per generation, and distributing the remaining survivors among the other FFs.
- The population per generation was kept constant at 40 parents and 100 children per generation.

The results were conclusive in that the FOA converged to the optimal solution in every case. Altering the parameters has the effect of altering the strategy for balancing diversification, ensuring that the entire solution space is explored, and intensification, exploiting the accumulated search experience. It also affects the particular strategies for achieving diversification, whether by employing more mutation or by selecting a majority of survivors according to the chosen FFs. The results show that the algorithm offers good results either by increasing diversification to facilitate optimization by exploring more possibilities, or by increasing intensification to facilitate optimization by building on good partial solutions, demonstrating the robustness of the approach.

Additional experiments, for which configuration details are omitted in sake of brevity, were conducted to investigate the solutions put forward by the FOA when the capacity of the MSB nodes embedded in the cell controllers was reduced, emulating an embedded device with limited resources. The results showed that the clients that were relocated from the embedded MSB node were skillfully affiliated. First of all, the conveyors were consistently relocated together with the line manager.
Fig. 5. Example flexible manufacturing system used to test the FOA.

Fig. 6. Chromosome structure of a population that converged to the Pareto-optimal solution.
client, which is in charge of scheduling and supervisory control of the material handling system. Next, robots were distributed in a way that balanced the load of the MSB nodes, but still grouping those that collaborate in a given work cell.

5.4. Analysis

The FOA proved to be robust when a hybrid selection strategy that favored solutions that were strong either for compromise fitness or specialty fitness. By keeping diversity of species in the chromosome population but simultaneously seeking chromosomes that compromised the different optimization criteria, the FOA is able to overcome local minima and converge to the Pareto minimum, which is the sought solution. This evolution strategy has also proven to be insensitive to changes in the parameters, and can therefore be expected to perform well for different cases.

Even though the message frequency and MSB capacity values were given based on typical values, these could be easily observed in real time and fed to the FOA for more accurate results. In order to observe these values, MSB nodes should merely count the number of messages of each type that are published, and periodically calculate throughput. Given that the accumulation operation is not computationally expensive, such an observation process would not add significant overhead.

The implementation of the FOA, which was not itself optimized to execute quickly, took no more than 3 s to execute runs of 1000 generations with populations of 40 parents and 100 children in a 2 GHz Pentium IV PC. This performance suggests that the FOA is suited for real-time use, and could easily be executed periodically every 30–60 s to rapidly adjust to changes in the system.

In assigning coefficients to the weighted sum FF, the most significant weight factors are given to the QoS fitness (FF3) and Loading Peaks fitness (FF5), followed by the

<table>
<thead>
<tr>
<th>Population size</th>
<th>Selection strategy FF1–6 speciation</th>
<th>FF7 weighted fitness</th>
<th>FF1–7 speciation and weighted fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 parents /60 children</td>
<td>Local minimum</td>
<td>Local minimum</td>
<td>Global minimum</td>
</tr>
<tr>
<td>40 parents /100 children</td>
<td>Local minimum</td>
<td>Local minimum</td>
<td>Global minimum</td>
</tr>
</tbody>
</table>

Fig. 7. 3D plot of a capricious two-dimensional function, exposing multiple local minima and a marked global minimum.
loading variance fitness (FF4). The aggregate processing
fitness and broker-to-broker processing FFs are given less
significance. The distance FF was not considered in the
experiment as no initial topology was given. However, this
scenario is much less demanding as a sub-optimal solution
is already given, and less iterations are needed.

The weighted optimization goal considering the multiple
dimensions is therefore strongly directed: to improve the
QoS for time-critical messages and to maintain feasible and
reasonable computational resource utilization. Although
the latter goal is primarily targeted at CPU load balancing
and avoiding MSB node overload, keeping the loading
factor at low values reduces the delivery delays of
messages. This conclusion can be inferred by considering
MSB nodes as a variation of the traditional queuing
problem. As a consequence, the favored goals can be
viewed as two conflicting dimensions of QoS optimization:
minimizing delay by reducing hops and minimizing delay
by reducing utilization at a node. Naturally, the first goal
can be achieved by grouping all clients at the same node,
which is in direct conflict with the second goal.

6. Conclusions

Scalable CAMX Publish/Subscribe Frameworks based
on distributed MSBs are faced with the challenge of
providing transparent topology optimization. This optimi-
zation problem is strictly NP-hard and characterized by
multiple, conflicting optimization criteria. Given that
deterministic approaches are impractical for solving this
type of problem, an Evolutionary Multi-Objective Optimi-
(zation (EMOO) algorithm was produced, denominated
Framework Optimization Algorithm (FOA).

The FOA proved to be robust in consistently finding the
Pareto-optimal solution in a small number of iterations
(1000). Convergence was achieved in less than 3 s, which
demonstrates that the FOA is well suited for rapid
framework reconfiguration in response to changes in the
system, as well as for periodical optimization runs every
few minutes. In order to overcome local minima, the
selection strategy is based on choosing for survival both
solutions that are either good in compromising the
different optimization criteria and solutions that excel in
particular criteria. The algorithm was tuned to favor
solutions that improve the QoS for time-critical messages
used for control purposes, but that are also feasible and
reasonable in the computational requirement imposed on
the distributed nodes.

The FOA is not only applicable to CAMX Frameworks
and could be used for other event-based manufacturing
systems that use Publish/Subscribe middleware with little
or no adaptation. Included in this type of systems are peer-
to-peer multi-agent systems such as Actor-based Assembly
Systems (ABAS) (Martinez Lastra, 2004), messaging
systems based on the WS-Eventing specification as proposed in SIRENA (Jammes and Smit., 2005), or IEC-


IPC, 2003. IPC-2501—Definition for Web-Based Exchange of XML Data. IPC, Northbrook, USA.


Ivan M. Delamer is with the Institute of Production Engineering at the Tampere University of Technology serving as Research Scientist for the Factory Automation Division. Mr. Delamer earned his Diploma in Electrical Engineering (“with honours”) from the Institute of Technology of Buenos Aires (ITBA). He is currently studying towards the degree of Doctor of Technology at the Tampere University of Technology. Previous to his current position at the Institute of Production Eng., Mr. Delamer carried research at the Manufacturing Research Center of the Georgia Institute of Technology (Atlanta, GA).

Jose L. Martinez Lastra joined the Institute of Production Engineering at the Tampere University of Technology in 1999, and became Full Professor of Factory Automation in 2006. Prof. Lastra earned his advanced degrees (MS “with distinction” and Dr.Tech. “with commendation”) in Automation Engineering from the Tampere University of Technology. His undergraduate degree (Ingeniero Técnico Industrial in Electrical Engineering) is from the Universidad of Cantabria (ETS de Ingenieros Industriales y de Telecomunicación, Spain). Previous to his current position at the Institute of Production Eng., Prof. Lastra carried research at the Departamento de Ingeniería Eléctrica y Energética (Santander, Spain), the Mathematics Department (Tampere, Finland), the Hydraulics and Automation Institute (Tampere, Finland) and the Mechatronics Research Lab. of the Massachusetts Institute of Technology (Cambridge, MA).