Science of Computer Programming 221 (2022) 102842

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Computing (optimal) embeddings of directed bigraphs

Alessio Chiapperini [a], Marino Miculan [a,1], Marco Peressotti [b,*]

[a] DMIF, University of Udine, Udine, Italy
[b] IMADA, University of Southern Denmark, Odense, Denmark

ARTICLE INFO

Article history:
Received 4 January 2021
Received in revised form 7 July 2022
Accepted 11 July 2022
Available online 25 July 2022

Keywords:
Graph rewriting systems
Bigraphs
Weighted bigraphs
Integer linear programming

ABSTRACT

Bigraphs and *bigraphical reactive systems* are a well-known meta-model successfully used for formalizing a wide range of models and situations, such as process calculi, service oriented architectures, multi-agent systems, biological systems, etc. A key problem in the theory and the implementations of bigraphs is how to compute *embeddings*, i.e., structure-preserving mappings of a given bigraph (the *pattern* or *guest*) inside another (the *target* or *host*).
In this paper, we present an algorithm for computing embeddings for *directed* bigraphs, an extension of Milner's bigraphs which take into account the request directions between controls and names. This algorithm solves the embedding problem by means of a reduction to a *constraint satisfaction problem*. We first prove soundness and completeness of this algorithm; then we present an implementation in jLibBig, a general Java library for manipulating bigraphical reactive systems. The effectiveness of this implementation is shown by several experimental results. Finally, we show that this algorithm can be readily adapted to find the *optimal* embeddings in a *weighted* variant of the embedding problem.

1. Introduction

Bigraphical Reactive Systems (BRSs) are a family of graph-based formalisms intended to be a meta-model for distributed, ubiquitous computing [26,33,36]. In this approach, system configurations are represented by *bigraphs*, data structures composed by two orthogonal graphs, a *place graph* and a *link graph*, describing the locations and the logical connections of (possibly nested) components, respectively. The dynamics of a system is defined by means of *graph rewriting rules*, which can replace and change components' positions and connections. Being a *meta*model, BRSs provide a range of general results and tools which can be readily instantiated with the specific model under scrutiny: libraries for bigraph manipulation [4,34,35], simulation tools [17,30,32,44], graphical editors [16], model checkers [39], modular composition [38], etc. Indeed, BRSs have been successfully used to formalize a wide range of models and situations, including process calculi, context-aware systems, web-service orchestration languages, cyber-physical scenarios, biological systems, software architectures, and IoT systems [2,3,6–8,29,36,41,47–50].

In the wake of these results, for covering further aspects of distributed systems several extensions of original Milner's bigraphs have been proposed. An example is *bigraphs with sharing* [43], which allow place graphs to be DAGs and not only trees. Another variant, which is the one we consider in this paper, is *directed bigraphs* [20,21], introduced to overcome the

* Corresponding author.
E-mail addresses: marino.miculan@uniud.it (M. Miculan), peressotti@imada.sdu.dk (M. Peressotti).
[1] Supported by Italian MIUR PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems).

https://doi.org/10.1016/j.scico.2022.102842

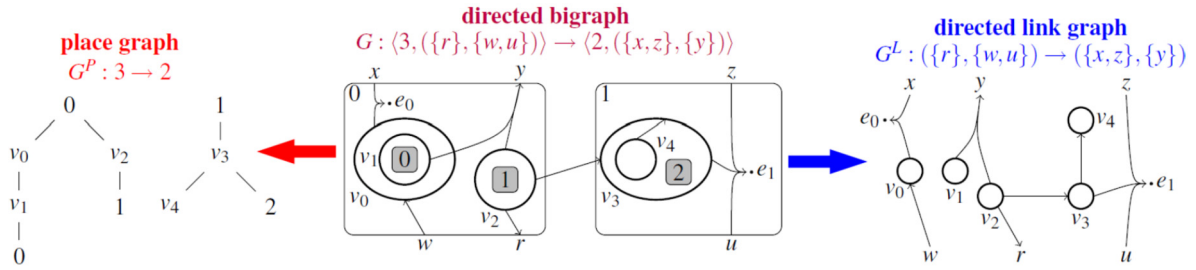


Fig. 1. An example of directed bigraph and its place and link graphs [22].

fact that in original bigraphs links can go only from "inside" to "outside": a component can connect to the surrounding environment, but it cannot access its subcomponents.[2] Directed bigraphs instead allow for *symmetric* link graphs, thus connections can go also "downward", as shown in the example of Fig. 1. A possible intuition is that these links represent the *access directions* of components to resources or other components; in this view, in Fig. 1 node v_2 accesses an external resource provided by the surrounding environment (through z), an internal resource provided by some subcomponent (through r), and also v_3 (in another location). As acknowledged by Milner in [36] "the mild extra complexity of directed bigraphs adds expressive power; indeed, the authors [of [20,21]] show how to encode the Fusion calculus of Parrow and Victor [37] which cannot be handled directly in bigraphs." In fact, directed bigraphs have been used to provide models of security protocols [19], molecular biology [3], access control [22], multi-agent systems [31], container-based systems [7], etc. Moreover, directed link graphs subsume also Sassone-Sobocinski's "input-linear bigraphs" [42], and preserve and generalize the important properties of original bigraphs, such as the existence of RPOs [20].

Another key notion is that of *bigraph embedding*. Informally, an embedding is a structure-preserving map from a bigraph (called *guest* or *pattern*) to another one (called *host* or *target*), similar to subgraph isomorphism. Embeddings are required for the application of rewriting rules (i.e., for matching a rule's redex inside an agent), but also for verifying properties on bigraphs in model-checking algorithms, for enforcing some well-formedness conditions over agents, etc. Despite the embedding problem is NP-complete in general [5], in most actual instances it turns out to be feasible. In fact, several algorithms have been proposed in literature for bigraphs with "traditional" link graphs; see e.g. [13,18,23,34,45,46].

In this work, we propose an algorithm for computing embedding of directed bigraphs. Rather than developing an *ad hoc* algorithm, we reduce the embedding problem to a *constraint satisfaction problem* (CSP). Our approach is *modular*, in the sense that we first solve the embedding of link graphs and of place graphs separately, with two different reductions (the former to a multi-flux problem, the latter to a bipartite matching). These two reductions produce two sets of constraints, which are then "glued" together by adding a small set of consistency constraints. This complete constraint set can be fed to a CSP solver, and from the solution(s) it provides we can reconstruct the sought embeddings.

This approach has many advantages. First, constraint satisfaction problems are well studied, and efficient solvers are available for many languages; for sake of definiteness in this paper we adopt Choco, a free open-source Java library for constraint programming [40], but in principle any modern CSP solver can be used. Second, since the embeddings of place and link graphs are reduced independently, we can easily port the algorithm to other versions of bigraphs by suitably adapting the corresponding part; e.g., we can specialize the algorithm to Milner's bigraphs or to input-linear bigraphs by simplifying constraints on the link graph embedding, but also bigraphs with sharing could be accommodated. Third, the use of constraint programming allows us to readily adapt the algorithm to *quantitative* versions of the embedding problem, where we look for *optimal* solutions according to a given notion of *embedding weight*. This problem has important applications, e.g., when we have to choose which rewriting rule to apply depending on the weight of the part to be replaced. Finally, reducing embedding to a constraint satisfaction problem allows us to consider also *approximate embeddings*, i.e., where we admit solutions which may not satisfy all constraints, just by solving the very same constraints using *approximate* CSP solvers [27].

The rest of the paper is structured as follows. In Section 2 we recall directed bigraphs and bigraphical reactive systems, slightly generalizing [7,20]. Then, the key notion of directed bigraph embedding is defined in Section 3. In Section 4 we present a reduction of the embedding problem for directed bigraphs to a constraint satisfaction problem (CSP) and show that it provides a sound and complete algorithm for computing embeddings. We have implemented this algorithm as an extension of jLibBig [35], a general Java library for BRSs; this implementation, with several experimental results, is reported in Section 5. In Section 6 we present a *weighted* version of the directed bigraph embedding problem, i.e., where embeddings are given a weight, and finding an embedding becomes an optimization problem; we show how our approach can be readily adapted to this case. In Section 7 we elaborate further this idea, by introducing two extensions of reactive systems over

[2] In pure bigraphs subcomponents can be connected by means of a mediating hyperedge, but this is not equivalent to a link going "inward", as edges are not subject to the place graph hierarchy and thus can be always seen as part of the surrounding environment. In directed bigraphs the mediating edge is an unnecessary artifact and can be omitted.

(directed) bigraphs: on one side, reactive systems over *weighted directed bigraphs*, on the other *weighted reactive systems* over directed bigraphs—and finally, their combination. Some conclusions and directions for future work are drawn in Section 8.

The main differences with the conference version of the current work [10,11] are a substantial expansion of Section 2 with several examples and discussion, of Section 5 with more detailed examples, and the addition of the new Sections 6 and 7 about weighted directed bigraphs.

2. Reactive systems on directed bigraphs

In this section we introduce a conservative extension of the notions of *directed* link graphs and bigraphs, and *directed bigraphical reactive systems*, originally defined in [20,21].

2.1. Directed bigraphs

Definition 1 *(Polarized interface).* A *polarized interface* X is a pair (X^-, X^+), where X^- and X^+, called *downward* and *upward* interface respectively, are two finite disjoint sets of names.

Definition 2 *(Polarized signature).* A *signature* is a pair $(\mathcal{K}, ar)$, where $\mathcal{K}$ is the set of *controls*, and $ar : \mathcal{K} \to \mathbb{N} \times \mathbb{N}$ is a map assigning to each control its polarized arity, that is, a pair $\langle n, m \rangle$ where n, m are the numbers of *positive* and *negative* ports of the control, respectively.

We define $ar^+, ar^- : \mathcal{K} \to \mathbb{N}$ as shorthand for the *positive* and *negative* ports of controls: $ar^+ \triangleq \pi_1 \circ ar$, $ar^- \triangleq \pi_2 \circ ar$.

The main difference between this definition and that from [20] is that we allow also for *inward*, here called "negative", ports in controls, whereas in [20], like in [36], controls have only outward ports. This turns up also in the definition of *points* and *handles* in the definition of link graphs.

Definition 3 *(Directed Link Graph).* A directed link graph $A : X \to Y$ is a quadruple $A = (V, E, ctrl, link)$ where V, E are the sets of vertexes and edges, X, Y are polarized interfaces and $ctrl : V \to \mathcal{K}$ is map assigning controls to vertexes, while the link map is a function $link : Pnt(A) \to Lnk(A)$ where

$$Prt^+(A) \triangleq \sum_{v \in V} ar^+(ctrl(v)) \qquad Prt^-(A) \triangleq \sum_{v \in V} ar^-(ctrl(v))$$
$$Pnt(A) \triangleq X^+ \uplus Y^- \uplus Prt^+(A) \qquad Lnk(A) \triangleq X^- \uplus Y^+ \uplus E \uplus Prt^-(A)$$

such that $link(Y^-) \cap Y^+ = \emptyset$.

The elements of $Pnt(A)$ and $Lnk(A)$ are called the *points* and the *handles* of A, respectively.

The constraint on *link* in this definition forbids a downward name of the outer interface to be connected to an upward name of the same interface. This guarantees that composition of link graphs (along the correct interfaces) is well defined.

Graphically, directed link graphs are depicted like ordinary link graphs but arrows are drawn on links and edges are explicitly represented as (placeless) dots.

Inward ("negative") ports add more flexibility to link graphs: an outward port of a node can be connected directly to an inner port of another node or even the same node (thus allowing also self-loops) without passing through an edge or a name. This mild extension enables us to represent more faithfully information like roles and asymmetric dependencies between entities in the model. Without directed links between nodes, a dependency would be represented by means of an extra node or edge, acting as a mediating proxy, as in the following example.

Example 4. Let us consider a bigraph signature for representing multi-tier client-server architectures, where clients can access web services and servers have access to some backend database management systems. Servers and DBMS have their own disk spaces. Moreover, server and clients share an access token (possibly known to other agents).

This situation can be modeled using a signature with four kinds of nodes: $client : \langle 2, 0 \rangle$, $server : \langle 3, 1 \rangle$, $db : \langle 1, 1 \rangle$, $disk : \langle 0, 1 \rangle$. The inward port of server nodes represents the services offered to clients; the inward port of db nodes represents the service offered to web servers, and the inward port of disk nodes represents the accessibility of these storage devices. An example is given in Fig. 2(a), where the directed link graph $A : \langle \emptyset, \emptyset \rangle \to \langle \{S\}, \{K\} \rangle$ represents a three-tier architecture with two clients, one server and one DBMS, each equipped with a disk storage. The clients access the service provided by the server, and the server accesses the service provided by the DBMS. Clients and web server share a key K provided by the environment. The service is offered also to other clients via the outer name S.

The same situation is rendered as a "standard" (i.e., non directed) link graph $B : \emptyset \to \{S, K\}$ in Fig. 2(b). In standard link graphs controls have only outward ports, and names in the outer interface cannot be connected to anything, but only be the target of the link map. So we have to add extra edges e_1, e_2, e_3 to mediate between WS and HD_1, $DBMS$ and HD_2, $DBMS$ and WS. These extra edges make these connections symmetric; hence we lose the difference between who provides the service and who uses it. Moreover, the direction of the name S had to be reversed, thus nullifying the difference between what is supplied to the environment (i.e., S) and what is provided by the environment (i.e., K).

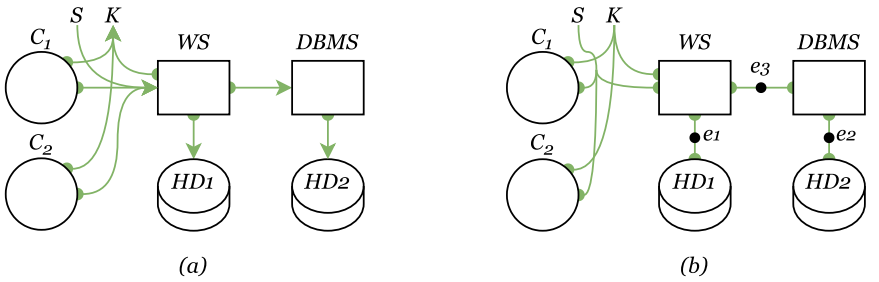


Fig. 2. (a) A directed link graph $A : \langle \emptyset, \emptyset \rangle \to \langle \{S\}, \{K\} \rangle$ representing two clients, one server and one DBMS (with their disk storages). (b) The same situation, rendered as a "standard" (i.e., pure, non directed) link graph $B : \emptyset \to \{S, K\}$.

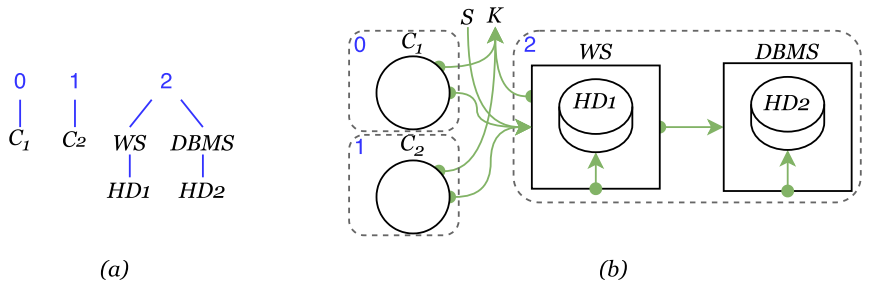


Fig. 3. (a) A place graph $F : 0 \to 3$. (b) A directed bigraph $G : \langle 0, \langle \emptyset, \emptyset \rangle \rangle \to \langle 3, \langle \{S\}, \{K\} \rangle \rangle$ representing a server and a backend DBMS on the same location, each including its own storage, and two clients on two other locations. This bigraph is obtained by combining the place graph aside with the link graph in Figure 2(a).

The definition of place graph is the same as pure bigraphs; we report it here, for sake of completeness.

Definition 5 *(Place Graph).* A place graph F is a triple $F = (V_F, ctrl_F, prnt_F) : m \to n$ where:

- m and n are finite ordinals representing respectively the inner and outer interfaces, they index the sites and roots of the place graph;
- $V_F \subset \mathcal{V}$ is a finite set of nodes;
- $ctrl_F : V_F \to \mathcal{K}$ is a control map assigning to each node a control belonging to signature $\mathcal{K}$;
- $prnt_F : m \uplus V_F \to V_F \uplus n$ is a parent map which is acyclic, meaning that if $prnt_F^i(v) = v$ for some $v \in V_F$, then $i = 0$.

Directed bigraphs are composed by a directed link graph and a place graph (cf. Fig. 1).

Definition 6 *(Directed Bigraph).* An *interface* $I = \langle m, X \rangle$ is composed by a finite ordinal m, called the *width*, and by a directed interface $X = (X^-, X^+)$.

Let $I = \langle m, X \rangle$ and $O = \langle n, Y \rangle$ be two interfaces; a *directed bigraph* with signature $\mathcal{K}$ from I to O is a tuple $G = (V, E, ctrl, prnt, link) : I \to O$ where

- I and O are the *inner* and *outer* interfaces;
- V and E are the sets of nodes and edges;
- $ctrl$, $prnt$, $link$ are the *control*, *parent* and *link* maps;

such that $G^L \triangleq (V, E, ctrl, link) : X \to Y$ is a directed link graph and $G^P \triangleq (V, ctrl, prnt) : m \to n$ is a place graph, that is, the map $prnt : m \uplus V \to n \uplus V$ is acyclic. The set of nodes and edges of G called *support of* G and is denoted by $|G|$. The bigraph G is denoted also as $\langle G^P, G^L \rangle$.

Example 7. Continuing Example 4, we can say that WS and $DBMS$ belong to the same location, while C_1, C_2 belong to two other locations. Moreover, we can say that HD_1 is an inner component of WS, and similarly HD_2 should be an inner part of $DBMS$. This structure is represented by the place graph in Fig. 3(a). Combining this place graph with the directed link graph in Fig. 2(a), we obtain the directed bigraph $G : \langle 0, \langle \emptyset, \emptyset \rangle \rangle \to \langle 3, \langle \{S\}, \{K\} \rangle \rangle$ in Fig. 3(b). Notice that in this bigraph, the links from WS to HD_1 and from $DBMS$ to HD_2 go "down" with respect to the order given by the place graph, i.e., from outside to inside; this is the opposite direction of standard bigraphs, where the links can go only "upward", i.e., from inside to outside.

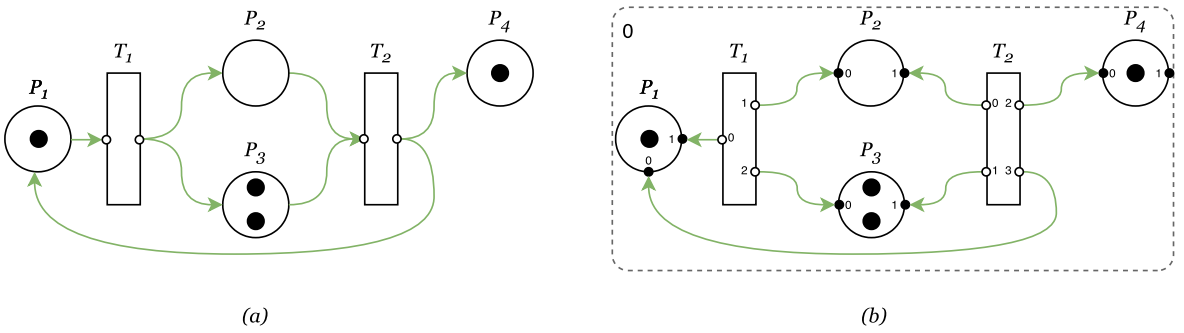


Fig. 4. (a) A Petri net, and (b) its representation as a directed bigraph.

We refer to [7] for a more elaborate application of directed bigraphs to the representation of processes, entities and components, especially in container-oriented architectures.

Example 8. In this example we sketch an encoding of Petri nets as directed bigraphs. The polarised signature has three controls: $\bullet : \langle 0, 0\rangle$ for tokens, $\circlearrowright : \langle 0, 2\rangle$ for places, and a countable family of controls $\square : \langle n, 0\rangle$ for transitions. The basic idea is that a place is an entity (an object) where tokens can be stored, and offering two services (two methods): port 0 is for putting tokens in the place, port 1 is for retrieving tokens from the place. A transition connects to one or the other ports, depending whether it consumes or produces tokens on that place. Fig. 4 shows an example encoding for a simple Petri net as a directed bigraph.

Two bigraphs may have the same structure and differ only for the choice of the concrete names of their internal components, i.e., by their support. Such bigraphs are called *support-equivalent*.

Definition 9. Two bigraphs $G : I \to O$ and $G' : I \to O$ are called *support-equivalent* if there is a bijection $\sigma : |G'| \to |G|$, called *support translation* that respects their structure in the sense that renaming the elements of the support of G according to σ yields G' i.e., $\sigma G = G'$.

This notion is needed for defining bigraphical reactive systems in terms of rewritings as we will see in a few paragraphs Definition 15.

Directed bigraphs can be composed along matching interfaces.

Definition 10 *(Composition and identities).* The composition of two place graphs $F : k \to m$ and $G : m \to n$, is defined in the same way as pure bigraphs (i.e., suitable grafting of forests).

If $F : X \to Y$ and $G : Y \to Z$ are two link graphs, their composition is the link graph $G \circ F \triangleq (V, E, ctrl, link) : X \to Z$ such that $V = V_F \uplus V_G$, $E = E_F \uplus E_G$, $ctrl = ctrl_F \uplus ctrl_G$, and $link : Pnt(G \circ F) \to Lnk(G \circ F)$ is defined as follows:

$$Pnt(G \circ F) = X^+ \uplus Z^- \uplus Prt^+(F) \uplus Prt^+(G)$$

$$Lnk(G \circ F) = X^- \uplus Z^+ \uplus Prt^-(F) \uplus Prt^-(G) \uplus E$$

$$link(p) \triangleq \begin{cases} prelink(p) & \text{if } prelink(p) \in Lnk(G \circ F) \\ link(prelink(p)) & \text{otherwise} \end{cases}$$

where $prelink : Pnt(G \circ F) \uplus Y^+ \uplus Y^- \to Lnk(G \circ F) \uplus Y^+$ is $link_F \uplus link_G$.

The identity link graph at X is $\mathsf{id}_X \triangleq (\varnothing, \varnothing, \varnothing_{\mathcal{K}}, \mathsf{id}_{X^- \uplus X^+}) : X \to X$.

If $F : I \to J$ and $G : J \to K$ are two bigraphs, their composite is

$$G \circ F \triangleq \langle G^P \circ F^P, G^L \circ F^L \rangle : I \to K$$

and the identity bigraph at $I = \langle m, X\rangle$ is $\langle \mathsf{id}_m, \mathsf{id}_{X^- \uplus X^+}\rangle$.

Definition 11 *(Juxtaposition).* For place graphs, the juxtaposition of two interfaces m_0 and m_1 is $m_0 + m_1$; the unit is 0. If $F_i = (V_i, ctrl_i, prnt_i) : m_i \to n_i$ are disjoint place graphs (with $i = 0, 1$), their juxtaposition is defined as for pure bigraphs.

For link graphs, the juxtaposition of two (directed) link graph interfaces X_0 and X_1 is $(X_0^- \uplus X_1^-, X_0^+ \uplus X_1^+)$. If $F_i = (V_i, E_i, ctrl_i, link_i) : X_i \to Y_i$ are two link graphs (with $i = 0, 1$), their juxtaposition is

$$F_0 \otimes F_1 \triangleq (V_0 \uplus V_1, E_0 \uplus E_1, ctrl_0 \uplus ctrl_1, link_0 \uplus link_1) : X_0 \otimes X_1 \to Y_0 \otimes Y_1$$

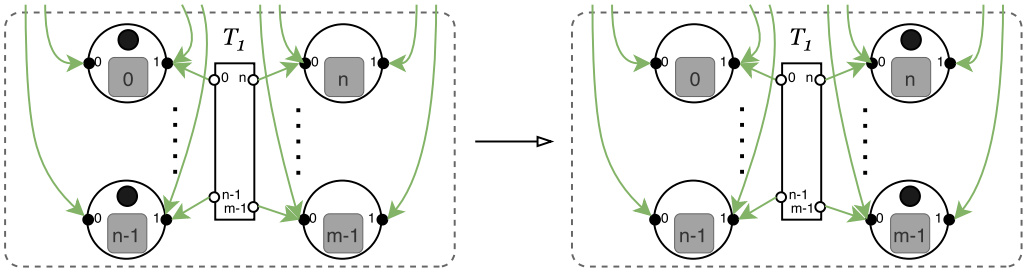


Fig. 5. Rewriting rule for a Petri net transition with n inputs and m outputs.

For bigraphs, the juxtaposition of two interfaces $I_i = \langle m_i, X_i \rangle$ (with $i = 0, 1$) is $\langle m_0 + m_1, (X_0^- \uplus X_1^-, X_0^+ \uplus X_1^+) \rangle$ (the unit is $\epsilon = \langle 0, (\varnothing, \varnothing) \rangle$). If $F_i \colon I_i \to J_i$ are two bigraphs (with $i = 0, 1$), their juxtaposition is

$$F_0 \otimes F_1 \triangleq \langle F_0^P \otimes F_1^P, F_0^L \otimes F_1^L \rangle \colon I_0 \otimes I_1 \to J_0 \otimes J_1.$$

Polarized interfaces and directed bigraphs over a given signature $\mathcal{K}$ form a monoidal category $\textsc{DBig}(\mathcal{K})$.

Milner's pure bigraphs [36] correspond precisely to directed bigraphs with positive interfaces only, and over signatures with only positive ports. On the other hand, one may wonder whether directed bigraphs can be obtained from pure ones. It turns out that, as pointed out by Debois [15], "directed bigraphs is the only variation of pure bigraphs which is not a sorting". In fact, directed bigraphs as per [20] can be obtained as a traced category over the category of pure bigraphs; however, we cannot properly represent controls with negative ports, as those used in the present paper, using controls with positive ports only like in [20,36].

2.2. Reactive systems over directed bigraphs

In order to define reactive systems over bigraphs, we need to define how a parametric reaction rule (i.e., a pair of "redex-reactum" bigraphs) can be instantiated. Essentially, in the application of the rule, the "sites" of the reactum must be filled with the parameters appearing in the redex. This relation can be expressed by specifying an *instantiation map* in the rule.

Definition 12. An *instantiation map* $\eta \colon \langle m, X \rangle \to \langle m', X' \rangle$ is a pair $\eta = (\eta^P, \eta^L)$ where

- $\eta^P : m' \to m$ is a function mapping sites of the reactum to sites of the redex; for each $j \in m'$, it determines that the j-th site of the reactum is filled with the $\eta(j)$-th parameter of the redex.
- $\eta^L \colon \left(\sum_{i=0}^{m'-1} X\right) \to X'$ is a wiring (i.e., a link graph without nodes nor edges), which is responsible for mapping names of the redex to names of the reactum. This can be described as a pair of functions $\eta^L = (\eta^+, \eta^-)$ where $\eta^+ \colon \left(\sum_{i=0}^{m'-1} X^+\right) \to X'^+$ and $\eta^- \colon X'^- \to \sum_{j=0}^{m'-1} X^-$.

We can now define the dynamics of directed bigraphs, starting with the formal definition of parametric reaction rules.

Definition 13 *(Parametric reaction rule).* A *parametric reaction rule* for bigraphs is a triple of the form $(R \colon I \to J, R' \colon I' \to J, \eta \colon I \to I')$ where R is the parametric redex, R' the parametric reactum and η is an instantiation map.

Example 14. Continuing Example 8, the generic rewriting rule for a transition with n inputs and m outputs is given in Fig. 5. Notice that there is a (downward) name for each port of each place, in order to allow other transitions to access these places. There is such a rule for each $n, m \in \omega$, yielding a countable set of rewriting rules. Of course, for a given Petri net, we can restrict to a finite set of rewriting rules, given by the transition with the largest number of connections.

We can now define the key notion of reactive systems over directed bigraphs, which is a generalization of that in [21,36]. Let $Ag(\mathcal{K})$ be the set of *agents* (i.e., bigraphs with no inner names nor sites) over a signature $\mathcal{K}$.

Definition 15. A *reactive system over directed bigraphs* (or simply bigraphical reactive systems) $BRS(\mathcal{K}, \mathcal{R})$ is defined by a signature $\mathcal{K}$ and a set $\mathcal{R}$ of parametric reaction rules. A reactive system $BRS(\mathcal{K}, \mathcal{R})$ induces a rewriting relation $\twoheadrightarrow \subseteq Ag(\mathcal{K}) \times Ag(\mathcal{K})$ according to the following rule:

$$\frac{(R, R', \eta) \in \mathcal{R} \qquad A = C \circ (\sigma R \otimes Id_Z) \circ \omega \circ (D_0 \otimes \ldots \otimes D_{m-1}) \\ A' = C \circ (\sigma' R' \otimes Id_Z) \circ \omega' \circ (D_{\eta^P(0)} \otimes \ldots \otimes D_{\eta^P(m'-1)})}{A \twoheadrightarrow A'} \tag{1}$$

where the support translations σ and σ' agree on $|R| \cap |R'|$, ω and ω' (called *wiring maps*) are defined as follows:

$$\omega\colon \sum_{i=0}^{m-1} X_i \to X \oplus Z \qquad \omega'\colon \sum_{j=0}^{m'-1} X_{\eta^P(j)} \to X' \oplus Z$$

$$\omega^+\colon \sum_{i=0}^{m-1} X_i^+ \to X^+ \uplus Z^+ \qquad \omega'^+\colon \sum_{j=0}^{m'-1} X^+_{\eta^P(j)} \to X'^+ \uplus Z^+$$

$$\omega^-\colon X^- \uplus Z^- \to \sum_{i=0}^{m-1} X_i^- \qquad \omega'^-\colon X'^- \uplus Z^- \to \sum_{j=0}^{m'-1} X^-_{\eta^P(j)}$$

$$\omega'^+(j,x) \triangleq \begin{cases} \eta^+(j, \omega^+(\eta(j),x)) & \text{if } \omega^+(\eta^P(j),x) \in X^+ \\ \omega^+(\eta^P(j),x) & \text{if } \omega^+(\eta(j),x) \in Z^+ \end{cases}$$

$$\omega'^-(x) \triangleq (j,y) \text{ for } j \in \eta^{P^{-1}}(i) \text{ and } (i,y) \in \eta^-(x)$$

The difference with respect to the previous versions of BRS is that now links can descend from the redex (and reactum) into the parameters, as it is evident from the fact that redexes and reactums in rules may have generic inner interfaces (I and I'). This is very useful for representing a request flow which goes "downwards", e.g. connecting a port of a control in the redex to a port of an inner component (think of, e.g., a linked library).

However, this poses some issues when the rules are not linear. If any of D_i's are cancelled by the rewriting, the controls in it disappear as well, and we may be not able to connect some name descending from R or Id_Z anymore. More formally, this means that the map ω^- can be defined only if for every $x \in (X'^- \uplus Z^-)$ there are j, y such that $(\eta^P(j), y) = \eta^-(x)$. We can have two cases:

1. for some x, there are no such j, y. This means that ω is not defined and hence the rule cannot be applied.
2. for each x, there are one or more pairs (j, y) such that $(\eta^P(j), y) = \eta^-(x)$. This means that for a given source agent decomposition, there can be several ways to define ω^-, each yielding a different application of the same rule.

Overall, the presence of downward names in parameters adds a new degree of non-determinism to Directed BRSs, with respect to previous versions of BRSs.

3. Directed bigraph embeddings

As we have seen in the previous section, to execute or simulate a BRS it is necessary to find the occurrences of a redex R within a given bigraph A. Definition 15 formalises the "finding the occurrences of a redex R in A" as a (directed) bigraph matching problem i.e., as the finding of a suitable decomposition of A. This presentation style is perhaps the most common in the literature of bigraphs and directed bigraphs and it allows for a succinct definition of rewriting. Højsgaard introduced in [24] the notion of *bigraph embeddings* as an alternative formulation of the occurrence problem, where occurrences of R in A are represented by suitable mappings from the components of R to those of A. As pointed out in [24], this formulation is more suitable for implementation since it contains enough information to subsume matches while dispensing from the cost of explicitly computing decomposition of the agent A into context, redex, and parameters. Indeed, the majority of the implementations of BRSs rely on embeddings in some form as they compute maps between concrete bigraphs [1,25,35,45]. In this section we extend the definition of embedding from [24] to the directed case with the notion of *directed bigraph embedding*.

Directed link graph Intuitively an embedding of link graphs is a structure preserving map from one link graph (the *guest*) to another (the *host*). This map contains a pair of injections: one for the nodes and one for the edges (i.e., a support translation). The remainder of the embedding map specifies how names of the inner and outer interfaces should be mapped into the host link graph. Outer names can be mapped to any link; here injectivity is not required since a context can alias outer names. Dually, inner names can be mapped to hyper-edges linking sets of points in the host link graph and such that every point is contained in at most one of these sets.

Definition 16 *(Directed link graph embedding).* Let $G : X_G \to Y_G$ and $H : X_H \to Y_H$ be two directed link graphs. A directed link graph embedding $\phi\colon G \hookrightarrow H$ is a map $\phi \triangleq \phi^v \uplus \phi^e \uplus \phi^i \uplus \phi^o$, assigning nodes, edges, inner and outer names with the following constraints:

(L1) $\phi^v\colon V_G \rightarrowtail V_H$ and $\phi^e\colon E_G \rightarrowtail E_H$ are injective;
(L2) $ctrl_G = ctrl_H \circ \phi^v$;

(L3) $\phi^i : Y_H^- \uplus X_H^+ \uplus P_H^+ \rightharpoonup X_G^+ \uplus Y_G^- \uplus P_G^+$ is a partial function s.t.:

$$\phi^i(x) \triangleq \begin{cases} \phi^{i^-}(x) & \text{if } x \in Y_H^- \uplus P_H^+ \\ \phi^{i^+}(x) & \text{if } x \in X_H^+ \uplus P_H^+ \end{cases} \quad \text{where} \quad \begin{array}{l} \phi^{i^-} : Y_H^- \uplus P_H^+ \rightharpoonup Y_G^- \uplus P_G^+ \\ \phi^{i^+} : X_H^+ \uplus P_H^+ \rightharpoonup X_G^+ \uplus P_G^+ \\ dom(\phi^{i^+}) \cap dom(\phi^{i^-}) = \varnothing \end{array}$$

(L4) $\phi^o : X_G^- \uplus Y_G^+ \rightharpoonup E_H \uplus X_H^- \uplus Y_H^+ \uplus P_H^-$ is a partial map s.t.:

$$\phi^o(y) \triangleq \begin{cases} \phi^{o^-}(y) & \text{if } y \in X_G^- \\ \phi^{o^+}(y) & \text{if } y \in Y_G^+ \end{cases} \quad \text{where} \quad \begin{array}{l} \phi^{o^-} : X_G^- \rightharpoonup E_H \uplus X_H^- \uplus P_H^- \\ \phi^{o^+} : Y_G^+ \rightharpoonup E_H \uplus Y_H^+ \uplus P_H^- \end{array}$$

(L5a) $img(\phi^e) \cap img(\phi^o) = \varnothing$;
(L5b) $\forall v \in V_G, \forall j \in ar(ctrl(v)) \;.\; \phi^i((\phi^v(v), j)) = \bot$;
(L6a) $\phi^p \circ link_G^{-1}|_{E_G} = link_H^{-1} \circ \phi^e$;
(L6b) $\forall v \in V_G, \forall i \in ar(ctrl(v)) \;.\; \phi^p \circ link_G^{-1}((v,i)) = link_H^{-1} \circ \phi^{port}((v,i))$;
(L7) $\forall p \in dom(\phi^i) : link_H(p) = (\phi^o \uplus \phi^e)(link_G \circ \phi^i(p))$.

where $\phi^p \triangleq \phi^{i^+} \uplus \phi^{o^-} \uplus \phi^{port}$ and $\phi^{port} : P_G \rightarrowtail P_H$ is $\phi^{port}(v,i) \triangleq (\phi^v(v), i)$.

The first three conditions are on the single sub-maps of the embedding. Conditions (L5a) and (L5b) ensure that no components (except for outer names) are identified; condition (L6a) imposes that points connected by the image of an edge are all covered. Finally, conditions (L2), (L6b) and (L7) ensure that the guest structure (i.e. node controls and point linkings) is preserved.

Place graph Like link graph embeddings, place graph embeddings are structure preserving, injective maps from nodes together with maps for the inner and outer interfaces. In particular, a site is mapped to the set of sites and nodes that are "put under it" and a root is mapped to the host root or node that is "put over it" splitting the host place graphs in three parts: the guest image, the context and the parameter (which are above and below the guest image).

Definition 17 *(Place graph embedding [24, Def 7.5.4]).* Let $G : n_G \to m_G$ and $H : n_H \to m_H$ be two place graphs. A place graph embedding $\phi : G \hookrightarrow H$ is a map $\phi \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{s}} \uplus \phi^{\mathsf{r}}$ (assigning nodes, sites and roots respectively) such that:

(P1) $\phi^{\mathsf{v}} : V_G \rightarrowtail V_H$ is injective;
(P2) $\phi^{\mathsf{s}} : n_G \rightarrowtail \wp(n_H \uplus V_H)$ is fully injective;
(P3) $\phi^{\mathsf{r}} : m_G \to V_H \uplus m_H$ in an arbitrary map;
(P4) $img(\phi^{\mathsf{v}}) \cap img(\phi^{\mathsf{r}}) = \emptyset$ and $img(\phi^{\mathsf{v}}) \cap \bigcup img(\phi^{\mathsf{s}}) = \emptyset$;
(P5) $\forall r \in m_G : \forall s \in n_G : prnt_H^* \circ \phi^{\mathsf{r}}(r) \cap \phi^{\mathsf{s}}(s) = \emptyset$;
(P6) $\phi^{\mathsf{c}} \circ prnt_G^{-1}\Big|_{V_G} = prnt_H^{-1} \circ \phi^{\mathsf{v}}$;
(P7) $ctrl_G = ctrl_H \circ \phi^{\mathsf{v}}$;
(P8) $\forall c \in n_G \uplus V_G : \forall c' \in \phi^{\mathsf{c}}(c) : (\phi^{\mathsf{f}} \circ prnt_G)(c) = prnt_H(c')$;

where $prnt_H^*(c) = \bigcup_{i<\omega} prnt^i(c)$, $\phi^{\mathsf{f}} \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{r}}$, and $\phi^{\mathsf{c}} \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{s}}$.

These conditions follow the structure of Definition 16, the main difference is (P5) which states that the image of a root cannot be the descendant of the image of another. Conditions (P1), (P2) and (P3) are on the three sub-maps composing the embedding; (P4) and (P5) ensure that no components are identified; (P6) imposes surjectivity on children and the last two conditions require the guest structure to be preserved by the embedding.

Directed bigraph Finally, a directed bigraph embedding can be defined as a pair composed by a directed link graph embedding and a place graph embedding, with a consistent interplay of these two structures. The interplay is captured by two additional conditions ensuring that points (resp. handles) in the image of guest upward (resp. downward) inner names reside in some parameter defined by the place graph embedding (i.e. descends from the image of a site).

Definition 18 *(Directed bigraph embedding).* Let $G : \langle n_G, X_G \rangle \to \langle m_G, Y_G \rangle$ and $H : \langle n_H, X_H \rangle \to \langle m_H, Y_H \rangle$ be two directed bigraphs. A directed bigraph embedding is a map $\phi : G \hookrightarrow H$ given by a place graph embedding $\phi^P : G^P \hookrightarrow H^P$ and a link graph embedding $\phi^L : G^L \hookrightarrow H^L$ subject to the following constraints:

(B1) $dom(\phi^{i^+}) \subseteq X_H^+ \uplus \{(v,i) \in P_H^+ \mid \exists s \in n_G, k \in \mathbb{N} : prnt_H^k(v) \in \phi^s(s)\}$;
(B2) $img(\phi^{o^-}) \subseteq X_H^- \uplus \{(v,i) \in P_H^- \mid \exists s \in n_G, k \in \mathbb{N} : prnt_H^k(v) \in \phi^s(s)\}$.

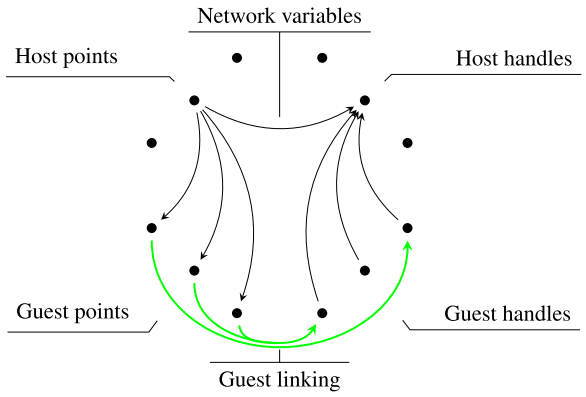


Fig. 6. Schema of the multi-flux network encoding.

4. Implementing the embedding problem in CSP

In this Section we present a constraint satisfaction problem that models the directed bigraph embedding problem. The encoding is based solely on integer linear constraints and is proven to be sound and complete.

Initially, we present the encoding for the directed link graph embedding problem and for the place graph embedding problem. Then we combine them providing some additional "gluing constraints" to ensure the consistency of the two sub-problems. The resulting encoding contains 37 constraint families (reflecting the complexity of the problem definition, see Section 3); hence we take advantage of the orthogonality of link and place structures for the sake of both exposition and adequacy proofs. We remark that the overall number of variables and constraints produced by the encoding is guaranteed to be polynomially bounded with respect to the size of the involved bigraphs, i.e., the number of nodes and edges.

4.1. Directed link graphs

Let us fix the guest and host bigraphs $G\colon X_G \to Y_G$ and $H\colon X_H \to Y_H$. We characterize the embeddings of G into H as the solutions of a suitable *multi-flux problem* which we denote as DLGE$[G, H]$. The main idea is to see the host points (i.e. positive ports, upward inner names and downward outer names) as sources, and the handles (i.e. edges, negative ports, upward outer names and downward inner names) as sinks (see Fig. 6). Each point outputs a flux unit and each handle inputs one unit for each point it links. Units flow towards each point handle following H edges and optionally taking a "detour" along the linking structure of the guest G (provided that some conditions about structure preservation are met). The formal definition of the flux problem is in Fig. 7.

The flux network reflects the linking structure and contains an edge connecting each point to its handle; these edges have an integer capacity limited to 1 and are represented by the variables defined in (3). The remaining edges of the network are organised in two complete biparted graphs: one between guest and host handles and one between guest and host points. Edges of the first sub-network are described by the variables in (2) and their capacity is bounded by the number of points linked by the host handle since this is the maximum acceptable flux and corresponds to the case where each point passes through the same hyper-edge of the guest link graph. Edges of the second sub-network are described by the variables in (4) and, like the first group of links, have their capacity limited to 1; to be precise, some of these variables will never assume a value different from 0 because guest points can receive flux from anything but the host ports (as expressed by constraint (10)). Edges for the link structure of the guest are presented implicitly in the flux preservation constraints (see constraint (8)). In order to fulfil the injectivity conditions of link embeddings, some additional *flux variables* (whereas the previous variables are *network variables*) are defined by (5). These are used to keep track and separate each flux on the bases of the points handle.

The constraint families (6) and (7) define the outgoing and ingoing flux of host points and handles respectively. The points have to send exactly one unit considering every edge they are involved with and the handles receive one unit for each of their point regardless if this unit comes from the point directly or from a handle of the guest.

The linking structure of the guest graph is encoded by the constraint family (8) which states that flux is preserved while passing through the guest i.e. the output of each handle has to match the overall input of the points it connects.

Constraints (9), (10), (19), (20), (21) and (22) shape the flux in the sub-network linking guest and host points. Specifically, (9) requires that each point from the guest receives at most one unit; this is needed when we want to be able to embed a redex where some points (e.g. upward inner names) would not match with an entity of the agent and (those points) would be deleted anyway when composing the resulting agent back. Constraints (10), (19) and (20) disable edges between guest ports and host inner names, between mismatching ports of matching nodes and between ports of mismatching nodes. Constraint (22) ensures that ascending inner names or descending outer names of the redex are not matched with positive

$$N_{h,h'} \in \{0, \dots, |link_H^{-1}(h')|\} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^- \tag{2}$$

$$N_{p,h'} \in \{0,1\} \qquad h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^-, p \in link_H^{-1}(h') \tag{3}$$

$$N_{p,p'} \in \{0,1\} \qquad p' \in X_G^+ \uplus P_G^+ \uplus Y_G^-, p \in X_H^+ \uplus P_H^+ \uplus Y_H^- \tag{4}$$

$$F_{h,h'} \in \{0,1\} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^- \tag{5}$$

$$\sum_k N_{p,k} = 1 \qquad p \in X_H^+ \uplus P_H^+ \uplus Y_H^- \tag{6}$$

$$\sum_k N_{k,h} = |link_H^{-1}(h)| \qquad h \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^- \tag{7}$$

$$\sum_k N_{h,k} = \sum_{p \in link_G^{-1}(h)} \sum_k N_{k,p} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^- \tag{8}$$

$$\sum_k N_{k,p} \le 1 \qquad p \in X_G^+ \uplus P_G^+ \uplus Y_G^- \tag{9}$$

$$N_{p,p'} = 0 \qquad p' \in P_G^+, p \in X_H^+ \uplus Y_H^- \tag{10}$$

$$\frac{N_{h,h'}}{|link_H^{-1}(h')|} \le F_{h,h'} \le N_{h,h'} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^-, link_G^{-1}(h) \neq \varnothing, link_H^{-1}(h') \neq \varnothing \tag{11}$$

$$N_{p,p'} \le F_{h,h'} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^-, p \in link_G^{-1}(h),\ p' \in link_H^{-1}(h') \tag{12}$$

$$F_{h,h'} \le \sum_{\substack{p \in link_G^{-1}(h) \\ p' \in link_H^{-1}(h')}} N_{p,p'} \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^-, link_G^{-1}(h) \neq \varnothing, link_H^{-1}(h') \neq \varnothing \tag{13}$$

$$\sum_k F_{h,k} = 1 \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^- \tag{14}$$

$$N_{p,h'} + F_{h,h'} \le 1 \qquad h \in E_G, h' \in E_H \uplus Y_H^+ \uplus X_H^- \uplus P_H^-, p \in link_H^{-1}(h') \tag{15}$$

$$F_{h,h'} + F_{h'',h'} + F_{h''',h'} + F_{h'^v,h'} \le 1 \qquad h \in E_G,\ h' \in Y_H^+ \uplus X_H^- \uplus P_H^-, h'' \in Y_G^+,\ h''' \in X_G^-,\ h'^v \in P_G^- \tag{16}$$

$$F_{h,h'} = 0 \qquad h \in E_G, h' \in Y_H^+ \uplus X_H^- \uplus P_H^- \tag{17}$$

$$F_{h,h'} \le 1 \qquad h \in E_G \uplus Y_G^+ \uplus X_G^- \uplus P_G^-, h' \in E_H \tag{18}$$

$$N_{p,p'} = 0 \qquad v \in V_G, v' \in V_H, ctrl_G(v) = ctrl_H(v) = c, i \neq i' \le c, p = (v,i) \in P_G^+ \uplus P_G^-, \ p' = (v',i') \in P_H^+ \uplus P_H^- \tag{19}$$

$$N_{p,p'} = 0 \qquad v \in V_G, v' \in V_H, ctrl_G(v) \neq ctrl_H(v), p = (v,i) \in P_G^+ \uplus P_G^-, p' = (v',i') \in P_H^+ \uplus P_H^- \tag{20}$$

$$\sum_{j \le c} N_{(v,j),(v',j)} = c \cdot N_{p,p'} \qquad v \in V_G, v' \in V_H, ctrl_G(v) = ctrl_H(v) = c, i \le c, p = (v,i) \in P_G^+ \uplus P_G^-, p' = (v',i') \in P_H^+ \uplus P_H^- \tag{21}$$

$$N_{p,p'} = 0 \qquad p \in P_H^+, p' \in X_G^+ \uplus Y_G^- \tag{22}$$

Fig. 7. Constraints of DLGE$[G, H]$.

ports of the agent. Finally, the flux of ports of the same node has to act compactly, as expressed by (21): if there is flux between the i-th ports of two nodes, then there should be flux between every other ports.

Constraints (11), (12) and (13) relate flux and network variables ensuring that the formers assume a true value if, and only, if there is actual flux between the corresponding guest and host handles. In particular, (12) propagates the information about the absence of flux between handles disabling the sub-network linking handles points and, *vice versa*, (13) propagates the information in the other way disabling flux between handles if there is no flux between their points.

The remaining constraints prevent fluxes from mixing. Constraint (14) requires guest handles to send their output to exactly one destination thus rendering the sub-network between handles a function assigning guest handles to host handles. This mapping is subject to some additional conditions when edges are involved: (17) and (18) ensure that the edges are injectively mapped to edges only, (16) forbids host outer names to receive flux from an edge and an outer name at the same time. Finally, (15) states that the output of host points cannot bypass the guest if there is flux between its handle and an edge from the guest.

Adequacy Let $\vec{N}$ be a solution of DLGE$[G, H]$. The corresponding link graph embedding $\phi\colon G \hookrightarrow H$ is defined as follows:

$$\phi^v(v) \triangleq v' \in V_H \text{ if } \exists i\colon N_{(v,i),(v',i)} = 1 \qquad \phi^e(e) \triangleq e' \in E_H \text{ if } F_{e,e'} = 1$$

$$\phi^i(x) \triangleq \begin{cases} \phi^{i^-}(x) & \text{if } x \in Y_H^- \uplus P_H^+ \\ \phi^{i^+}(x) & \text{if } x \in X_H^+ \uplus P_H^+ \end{cases} \qquad \phi^o(y) \triangleq \begin{cases} \phi^{o^-}(y) & \text{if } y \in X_G^- \\ \phi^{o^+}(y) & \text{if } y \in Y_G^+ \end{cases}$$

where

$$\phi^{o^-}(y) \triangleq y' \in X_H^- \uplus P_H^- \text{ if } F_{y,y'} = 1 \qquad \phi^{o^+}(y) \triangleq y' \in Y_H^+ \uplus P_H^- \text{ if } F_{y,y'} = 1,$$
$$\phi^{i^-}(x) \triangleq x' \in Y_G^- \uplus P_G^+ \text{ if } N_{x,x'} = 1, \qquad \phi^{i^+}(x) \triangleq x' \in X_G^+ \uplus P_G^+ \text{ if } N_{x,x'} = 1$$
$$\text{and } dom(\phi^{i^+}) \cap dom(\phi^{i^-}) = \varnothing.$$

It is easy to check that these components of ϕ are well-defined and compliant with Definition 16.

On the other way around, let $\phi\colon G \hookrightarrow H$ be a link graph embedding. The corresponding solution $\vec{N}$ of $DLGE[G, H]$ is defined as follows:

$$N_{p,p'} \triangleq \begin{cases} 1 & \text{if } p \in X_H^+ \uplus Y_H^- \wedge p' = \phi^i(p) \\ 1 & \text{if } p' = (v, i) \in P_G^+ \wedge p = (\phi^v(v), i) \\ 0 & otherwise \end{cases}$$

$$N_{p,h'} \triangleq \begin{cases} 1 & \text{if } h' = link_H(p) \wedge \nexists p'\colon N_{p,p'} = 1 \\ 0 & otherwise \end{cases}$$

$$N_{h,h'} \triangleq \begin{cases} 1 & \text{if } h' \in E_H \wedge h \in E_G \wedge h' = \phi^e(h) \\ 1 & \text{if } h' \in Y_H^+ \uplus X_H^- \wedge h \in Y_G^+ \uplus X_G^- \wedge h' = \phi^o(h) \\ 1 & \text{if } h = (v, i) \in P_G^- \wedge h' = (\phi^v(v), i) \\ 0 & otherwise \end{cases}$$

$$F_{h,h'} = 1 \overset{\triangle}{\Longleftrightarrow} N_{h,h'} \neq 0$$

Every constraint of $DLGE[G, H]$ is satisfied by the solution just defined.

The constraint satisfaction problem in Fig. 7 is sound and complete with respect to the directed link graph embedding problem given in Definition 16.

Proposition 19 *(Adequacy of* DLGE*). For any two concrete directed link graphs G and H, there is a bijective correspondence between the directed link graph embeddings of G into H and the solutions of* DLGE$[G, H]$.

4.2. Place graphs

Let us fix the guest and host place graphs: $G\colon n_G \to m_G$ and $H\colon n_H \to m_H$. We characterize the embeddings of G into H as the solutions of the constraint satisfaction problem in Fig. 8. The problem is a direct encoding of Definition 17 as a matching problem presented, as usual, as a bipartite graph. Sites, nodes and roots of the two place graphs are represented as nodes and partitioned into the guest and the host ones. For convenience of exposition, the graph is complete.

Edges are modelled by the boolean variables defined in (23); these are the only variables used by the problem. So far, a solution is nothing more than a relation between the components of guest and host containing only those pairs connected by an edge assigned a non-zero value. To capture exactly those assignments that are actual place graph embeddings some conditions have to be imposed.

Constraints (24) and (25) prevent roots and sites from the host to be matched with nodes or sites and nodes or roots respectively. (26) disables matching between nodes decorated with different controls. Constraint (27) prevents any matching for host nodes under a passive context (i.e. have an ancestor labelled with a passive control). (28) propagates the matching along the parent map from children to parents. Constraints (29) and (30) ensure that the matching is a function when restricted to guest nodes and roots (the codomain restriction follows by (24) and (25)). (31) says that if a node from the host cannot be matched with a root or a node/site from the guest at the same time; moreover, if the host node is matched with a node then it cannot be matched to anything else.

The remaining constraints are the counterpart of (28) and propagate matchings from parents to children. (32) applies to matchings between nodes and says that if parents are matched, then children from the host node are covered by children from the guest node. In particular, the matching is a perfect assignment when restricted to guest children that are nodes (because of (31)) and is a surjection on those that are sites. (33) imposes a similar condition on matchings between guest roots and host nodes. Specifically, it says that the matching has to cover child nodes from the guest (moreover, it is injective on them) leaving child sites to match whatever remains ranging from nothing to all unmatched children. Finally, (34) prevents matching from occurring inside a parameter.

Adequacy Let $\vec{M}$ be a solution of PGE$[G, H]$. The corresponding place graph embedding $\phi\colon G \hookrightarrow H$ is defined as follows:

$$\phi^{\mathsf{v}}(g) \triangleq h \in V_H \text{ if } \exists i\colon M_{h,g} = 1 \qquad \phi^{\mathsf{s}}(g) \triangleq \{h \in n_h \uplus V_H \mid M_{h,g} = 1\} \qquad \phi^{\mathsf{r}}(g) \triangleq h \in m_H \uplus V_H \text{ if } M_{h,g} = 1$$

$$M_{h,g} \in \{0,1\} \qquad g \in n_G \uplus V_G \uplus m_G, h \in n_H \uplus V_H \uplus m_H \tag{23}$$

$$M_{h,g} = 0 \qquad g \in n_G \uplus V_G, h \in m_H \tag{24}$$

$$M_{h,g} = 0 \qquad g \in V_G \uplus m_G, h \in n_H \tag{25}$$

$$M_{h,g} = 0 \qquad g \in V_G, h \in V_H, \mathit{ctrl}_G(g) \neq \mathit{ctrl}_H(h) \tag{26}$$

$$M_{h,g} = 0 \qquad g \in m_G, h \notin m_H, v \in \mathit{prnt}_H^*(h) \cap V_G, \mathit{ctrl}_G(v) \notin \Sigma_a \tag{27}$$

$$M_{h,g} \leq M_{h',g'} \qquad g \notin m_G, g' \in \mathit{prnt}_G(g), h \notin m_H, h' \in \mathit{prnt}_H(h) \tag{28}$$

$$\sum_{h \in V_H \uplus m_H} M_{h,g} = 1 \qquad g \in m_G \tag{29}$$

$$\sum_{h \in n_H \uplus V_H} M_{h,g} = 1 \qquad g \in V_G \tag{30}$$

$$m_G \cdot \sum_{g \in n_G \uplus V_G} M_{h,g} + \sum_{g \in m_G} M_{h,g} \leq m_G \qquad h \in V_H \tag{31}$$

$$|\mathit{prnt}_H^{-1}(h)| \cdot M_{h,g} \leq \sum_{\substack{h' \in \mathit{prnt}_H^{-1}(h),\\ g' \in \mathit{prnt}_G^{-1}(g)}} M_{h',g'} \qquad g \in V_G, h \in V_H \tag{32}$$

$$|\mathit{prnt}_G^{-1}(g) \setminus n_G| \cdot M_{h,g} \leq \sum_{\substack{h' \in \mathit{prnt}_H^{-1}(h) \setminus n_h,\\ g' \in \mathit{prnt}_G^{-1}(g) \setminus n_g}} M_{h',g'} \qquad g \in m_G, h \in V_H \tag{33}$$

$$M_{h,g} + \sum_{h' \in \mathit{prnt}_H^*(h), g' \in m_G} M_{h',g'} \leq 1 \qquad g \in V_G, h \in V_H \tag{34}$$

Fig. 8. Constraints of PGE$[G, H]$.

These components of ϕ are well-defined and compliant with Definition 17.

On the opposite direction, let $\phi\colon G \hookrightarrow H$ be a place graph embedding. The corresponding solution $\vec{M}$ of PGE$[G, H]$ is defined as follows.

$$M_{h,g} \triangleq \begin{cases} 1 & \text{if } g \in V_G \wedge h = \phi^{\mathsf{v}}(g) \\ 1 & \text{if } g \in m_G \wedge h = \phi^{\mathsf{r}}(g) \\ 1 & \text{if } g \in n_G \wedge h \in \phi^{\mathsf{s}}(g) \\ 0 & \text{otherwise} \end{cases}$$

It is easy to check that every constraint of PGE$[G, H]$ is satisfied by this solution. Hence, the constraint satisfaction problem in Fig. 8 is sound and complete with respect to the place graph embedding problem (Definition 17).

Proposition 20 *(Adequacy of* PGE*). For any two concrete place graphs G and H, there is a bijective correspondence between the place graph embeddings of G into H and the solutions of* PGE$[G, H]$.

4.3. Bigraphs

Let $G\colon \langle n_G, X_G\rangle \to \langle m_G, Y_G\rangle$ and $H\colon \langle n_H, X_H\rangle \to \langle m_H, Y_H\rangle$ be two bigraphs. By taking advantage of the orthogonality of the link and place structures we can define the constraint satisfaction problem capturing bigraph embeddings by simply composing the constraints given above for the link and place graph embeddings and by adding four consistency constraints to relate the solutions of the two problems. These additional constraint families are reported in Fig. 9. The families (35) and (36) ensure that solutions for DLGE$[G, H]$ and PGE$[G, H]$ agree on nodes since the map ϕ^{v} has to be shared by the corresponding link and place embeddings. The families (37) and (38) respectively, ensure that positive ports (negative ports resp.) are in the same image as upward inner names (downward inner names resp.) only if their node is part of the parameter i.e. only if it is matched to a site from the guest or it descends from a node that is so.

Conditions (37) and (38) correspond exactly to (B1) and (B2). It thus follows from Propositions 19 and 20 that the CSP defined by Figs. 7 to 9 is sound and complete with respect to the bigraph embedding problem given in Definition 18.

Theorem 21 *(Adequacy of* DBGE*). For any two concrete bigraphs G and H, there is a bijective correspondence between the bigraph embeddings of G into H and the solutions of* DBGE$[G, H]$.

$$M_{v,v'} = N_{p,p'} \qquad v \in V_H, v' \in V_G, p = (v,k) \in P_H^+, p' = (v',k) \in P_G^+ \tag{35}$$

$$M_{v,v'} = F_{h,h'} \qquad v \in V_H, v' \in V_G, h \in P_G^-, h' \in P_H^- \tag{36}$$

$$\sum_{p' \in X_G^+} N_{p,p'} \leq \sum_{\substack{h \in prnt_H^*(v) \\ g \in n_G}} M_{h,g} \qquad v \in V_H,\ p = (v,k) \in P_H^+ \tag{37}$$

$$\sum_{h \in X_G^-} F_{h,h'} \leq \sum_{\substack{h \in prnt_H^*(v) \\ g \in n_G}} M_{h,g} \qquad v \in V_H,\ h' = (v,k) \in P_H^- \tag{38}$$

Fig. 9. Constraints of DBGE[G, H].

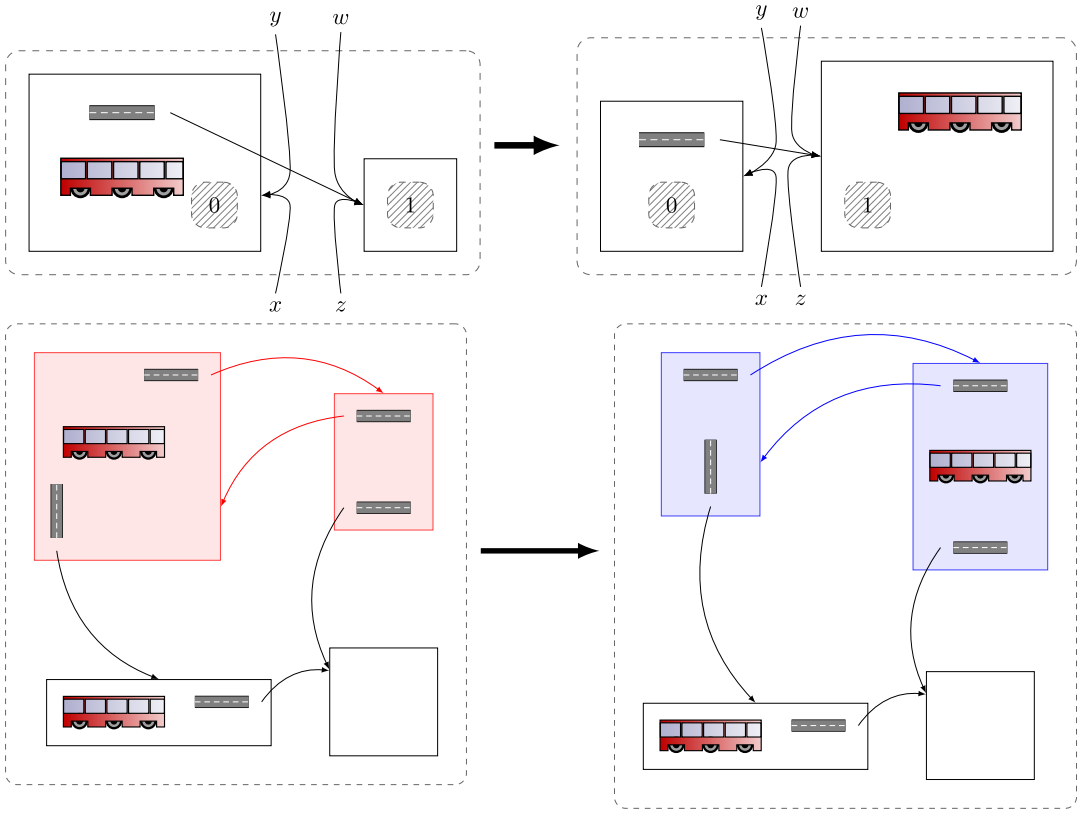


Fig. 10. Rewriting rule for the test cases (top) and an example of its application (bottom) where the embedding of the redex and the instantiation of the reactum to replace it are highlighted.

5. Experimental results

The reduction algorithm presented in the previous section has been successfully integrated into jLibBig, an extensible Java library for manipulating bigraphs and bigraphical reactive systems which can be used for implementing a wide range of tools and can be adapted to support several extensions of bigraphs [35]. The proposed algorithm is implemented by extending the data structures and the models for pure bigraphs to suit our definition of directed bigraphs.

In this section we test our implementation by simulating a system in which we want to track the position and the movements of a fleet of vehicles inside a territory divided in “zones”, which are accessible via “roads”. The rewriting rule and an example agent can be found in Fig. 10.

We consider two main application scenarios:

1. in the first, all possible reactions are explored;
2. in the second, only one reaction is explored.

We evaluate the running time of the different components of our algorithm: model construction, CSP resolution, building of the actual embedding and execution of the rewriting rule. Moreover, we want to evaluate how these performances

scale while increasing the size of the agent. The parameters used to build the tests are: number of zones, number of cars and "connectivity degree". The latter is a number between 1 and 100 representing the probability of the existence of a connection, where 100 means that every node is connected to all its neighbours.

We consider the following kinds of tests:

1. varying number of cars, with fixed number of zones and connectivity degree;
2. varying number of zones, with fixed number of cars and connectivity degree;
3. varying connectivity degree, with fixed numbers of zones and cars.

Each test case is made up of four groups of instances, where for each group we choose an increasing value for their fixed parameters. For each group we choose ten values for its variable parameter. The instances generation works as follows: for each test case and for each group of that particular test case we generate ten random instances for each combination of the values of the fixed parameters and the variable parameter. We then take the average of the running times of those ten random instances. At the end of the process, for each group we have tested 100 instances, 10 for each value of the variable parameter, so 400 instances for each test case and 1200 in total. To avoid the JVM warm-up effect (class loading, bytecode interpretation) and attain a realistic condition for the heap, we perform ten additional instances of each case. Measurements are taken by instrumenting jLibBig and using millisecond precision. All tests have been performed on an Intel Core i7-4710HQ (4 cores at 3.5GHz), 8 GB of RAM running on ArchLinux with kernel 5.5.2 and using OpenJDK 12; jLibBig v0.0.4 with Choco v4.10 as the underlying solver. Although the CPU used in the experiments is multicore, all steps under test (model construction, resolution, embedding, and rewriting) are sequential. Results for all tests are reported in Appendix A, Figs. A.16–A.24. The code for generating and running the experiment instances is available at [12].

Overall, we observe that of the four test phases, searching for a solution to the model and translating a solution into the structures used by jLibBig to represent an embedding are the most costly (cf. Fig. 11). The remaining two phases, namely, setting up the solver and applying a rewriting rule given the embedding, are negligible. While the cost of translating a solution into an embedding is essentially constant for instances of the same size (cf. Figs. 11b and 11c), the cost of finding the solutions reduces drastically after the first one (cf. Fig. 11d). All these observations are within our expectations.

In the remainder of this section, we review the results obtained in each kind of test.

Time vs. number of cars In this case we evaluate how our implementation scales with an increasing number of cars; see Fig. 11. We can observe that the total execution time increases linearly following to the number of solutions. In fact, the average execution time per solution is constant. We can also see that the two phases that contribute the most to the total running time are the solving phase of the CSP and the phase in which the embedding is built from the solution of the CSP. On the other hand, the building and rewriting phases are executed nearly instantly. We observe that the time needed to build the CSP is almost constant, while the rewriting time scales almost linearly as well.

Time vs. number of zones In this case we evaluate how our implementation scales with an increasing number of zones; see Fig. 12. We can see that the running time grows exponentially compared to the size of the network, especially the resolution time. Similarly to the previous test case, the time spent building the CSP and applying the reaction rule is negligible even though we can see that the time necessary to build the CSP increases linearly with the grid size. We can also observe that there is no correlation between the rewriting time and the number of zones.

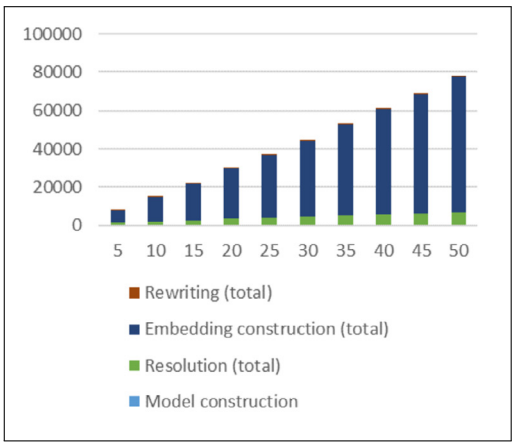
Time vs. connectivity degree In this case we evaluate how our implementation scales with an increasing connectivity degree; see Fig. 13. We can see that the running time scales exponentially, no matter the grid size or the number of cars. Differently from the previous cases, time is mainly consumed translating a solution into an embedding as shown by the breakdown of the average time per solution the translation in Fig. 13c. Once again, we see that although increasing, the time spent building the model and applying the rewriting rule is negligible.

6. Optimal directed embeddings

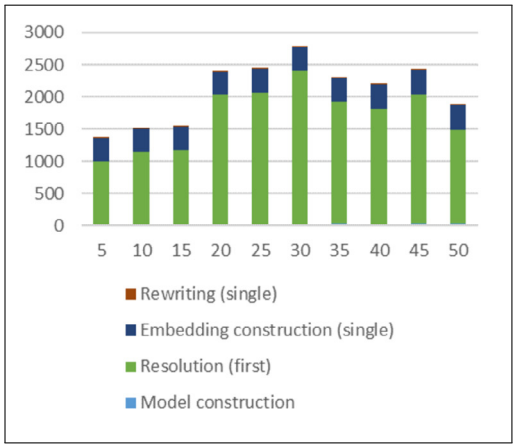
In this section we consider the optimisation problem for a *quantitative* variant of bigraph embedding, and show that our algorithm can be readily adapted to this setting.

Let us fix a semiring of *weights* $\mathbb{W}$. Given a guest G and a host H, one could consider a function assigning a value in $\mathbb{W}$ to each embedding of G into H; then, we can formulate an optimisation analogue of the embedding problem by looking for solutions of minimal (resp. maximal) weight. This problem has important applications; e.g., one can model reconfigurations of a system architecture by means of bigraphical rewritings [7], then the optimal embedding would be the one which minimizes the total cost of the components to be refactored. As another example, one could adapt the use case from Section 5 to penalise roads with pay tolls or heavy traffic.

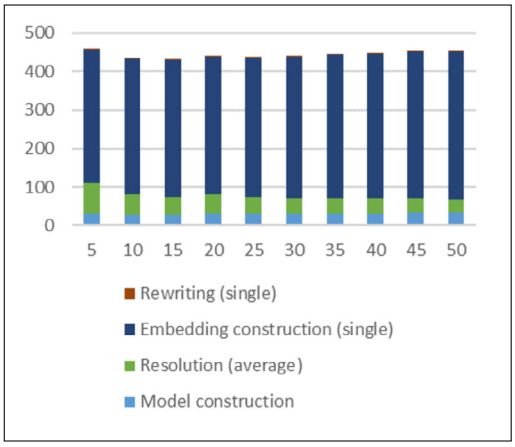
In our approach, we can readily reduce the optimal bigraph embedding problem to a (linear) constraint optimisation problem using the same model introduced in Section 4. To this end, we need to refine the notion of weighting function to meet the linearity requirement.



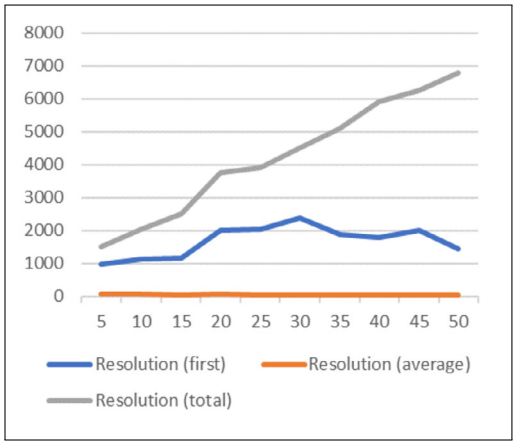
(a) First scenario, computing all reactions. Execution time (ms) vs. number of cars.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. number of cars.



(c) Mean time for single reaction. Execution time (ms) vs. number of cars.



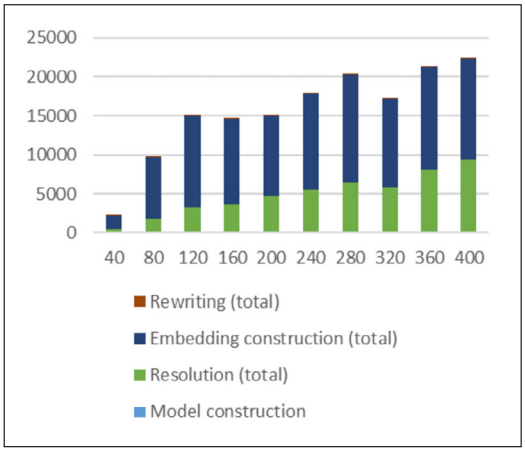
(d) Solution search. Execution time (ms) vs. number of cars.

Cars	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
5	19	30.3	977.7	82	1519.0	346.3	6440.5	0.376	7.0
10	37	28.3	1119.6	55	2026.3	350.3	12996.8	0.418	15.5
15	55	29.1	1154.6	46	2492.4	354.2	19340.0	0.377	20.6
20	73	30.9	1998.6	51	3756.1	357.8	26231.4	0.383	28.1
25	91	31.7	2037.1	43	3901.6	362.4	32910.0	0.412	37.4
30	109	30.3	2380.5	41	4490.5	366.0	39861.0	0.421	45.8
35	128	32.0	1883.7	40	5095.0	372.4	47627.7	0.399	51.0
40	146	31.7	1785.3	40	5921.4	375.6	54981.9	0.419	61.3
45	163	32.3	2007.8	38	6260.4	379.6	61916.6	0.414	67.5
50	184	32.4	1456.0	37	6795.7	383.3	70534.7	0.420	77.2

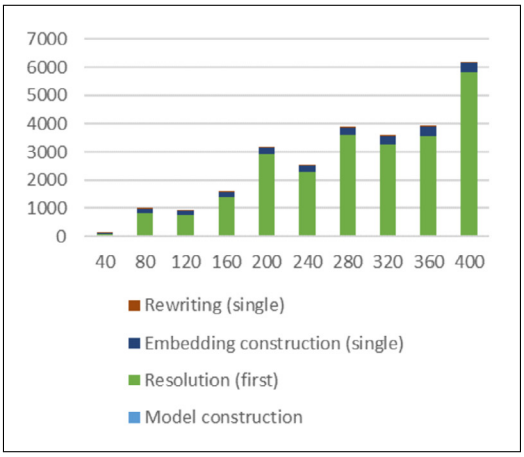
Fig. 11. Execution time (ms) vs. number of cars, 11x11 grid with 100% connectivity (time values are in milliseconds).

Given bigraphs G and H, we call $\mathbb{W}$-*weighting for* $(G \hookrightarrow H)$ any function χ that can be decomposed as functions with the following types:

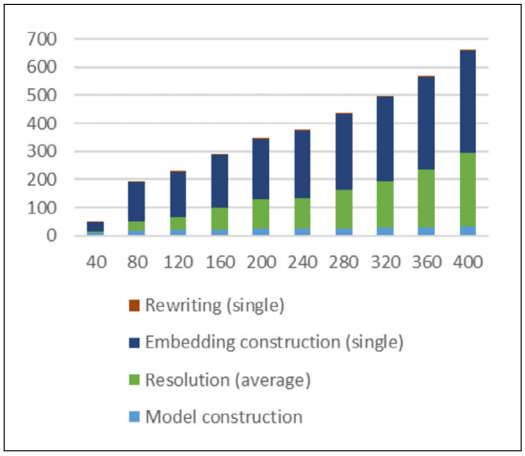
$$\chi^{\mathsf{v}} \colon V_G \times V_H \to \mathbb{W}$$
$$\chi^{\mathsf{s}} \colon n_G \times (n_H \uplus V_H) \to \mathbb{W}$$
$$\chi^{\mathsf{r}} \colon m_G \times (V_H \uplus m_H) \to \mathbb{W}$$
$$\chi^{\mathsf{e}} \colon E_G \times E_H \to \mathbb{W}$$



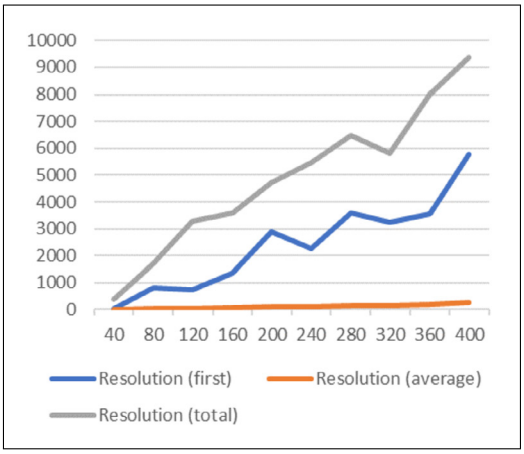
(a) First scenario, computing all reactions. Execution time (ms) vs. grid size.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. grid size.



(c) Mean time for single reaction. Execution time (ms) vs. grid size.



(d) Solution search. Execution time (ms) vs. grid size.

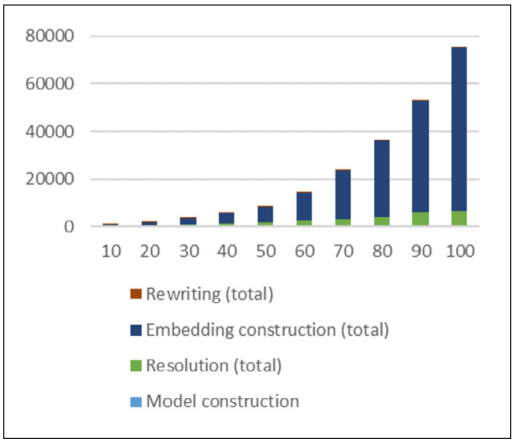
Zones	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
40	55	8.7	53.5	6.8	377.2	31.6	1749.1	0.190	10.5
80	56	18.3	797.7	31.8	1706.6	141.7	7905.7	0.292	16.3
120	72	20.6	724.6	46.0	3266.0	163.1	11770.0	0.310	22.4
160	59	21.4	1365.1	77.1	3590.1	187.9	11018.1	0.346	20.3
200	49	23.8	2900.1	106.8	4709.5	213.8	10370.4	0.313	15.2
240	51	24.9	2253.3	109.2	5470.1	239.7	12321.4	0.407	20.9
280	51	26.9	3580.2	139.0	6484.0	268.9	13725.7	0.413	21.1
320	38	28.4	3238.7	166.2	5822.3	297.9	11355.9	0.425	16.2
360	40	30.5	3548.0	205.0	8000.6	329.8	13249.4	0.460	18.5
400	36	31.6	5774.6	265.6	9364.3	361.6	12897.0	0.451	16.1

Fig. 12. Execution time (ms) vs. grid size, 70 cars and 100% connectivity (values are in milliseconds).

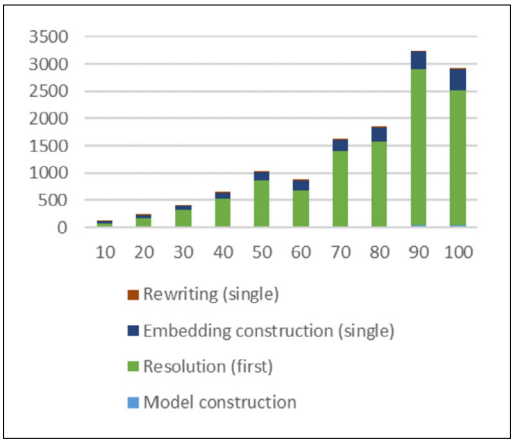
$$\chi^{\mathsf{i}} \colon (Y_H^- \uplus X_H^+ \uplus P_H^+) \times (Y_G^- \uplus X_G^+ \uplus P_G^+) \to \mathbb{W}$$

$$\chi^{\mathsf{o}} \colon (X_G^- \uplus Y_G^+) \times (E_H \uplus X_H^- \uplus Y_H^+ \uplus P_H^-) \to \mathbb{W}$$

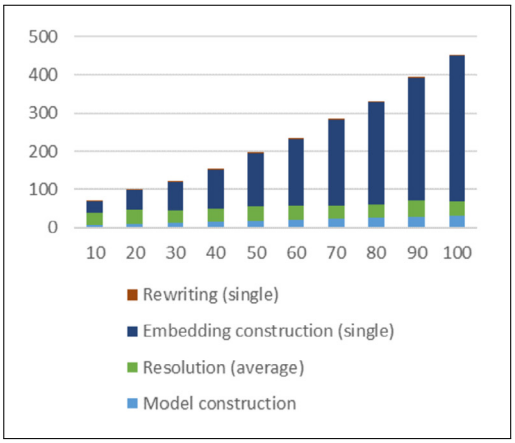
The function χ can be seen as a biparted graph whose edges are labelled in $\mathbb{W}$ and connect elements of G to elements of H coherently with the domains and codomains of the maps that define embeddings of G into H (e.g., there are no edges



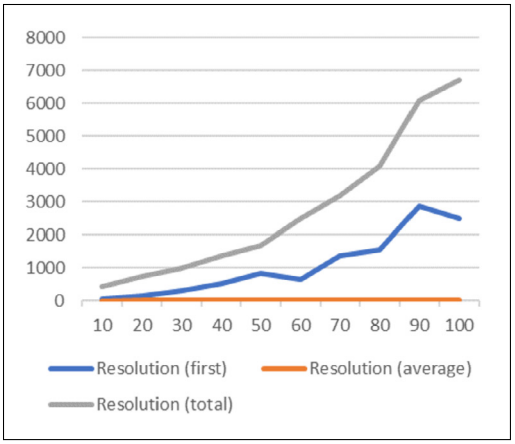
(a) First scenario, computing all reactions. Execution time (ms) vs. connectivity.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. connectivity.



(c) Mean time for single reaction. Execution time (ms) vs. connectivity.



(d) Solution search. Execution time (ms) vs. connectivity.

Connectivity	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
10%	15	8.6	66.4	29.7	429.3	29.9	445.7	0.177	2.6
20%	22	11.2	155.7	37.0	734.0	49.4	1112.1	0.211	4.7
30%	32	13.7	312.7	32.3	1006.6	74.5	2413.4	0.249	8.0
40%	43	16.1	514.6	33.1	1374.1	102.7	4401.2	0.265	11.3
50%	47	18.5	851.7	37.0	1681.0	139.0	6567.1	0.314	14.8
60%	67	20.8	661.7	36.9	2495.3	175.4	11772.6	0.332	22.3
70%	91	23.7	1374.9	35.6	3168.5	223.6	20418.4	0.370	33.6
80%	119	25.6	1546.4	34.3	4076.8	269.2	31954.1	0.385	45.7
90%	145	29.0	2877.4	42.1	6097.5	321.5	46683.0	0.406	58.9
100%	180	32.1	2485.8	37.3	6720.0	380.9	68634.2	0.435	78.3

Fig. 13. Execution time (ms) vs. connectivity, 11x11 grid and 70 cars (values are in milliseconds).

between a node and an outer name). Given a weighting χ for G and H and an embedding $\phi\colon G \hookrightarrow H$, we can assign a weight $\chi \bullet \phi$ to ϕ by laying the graph of ϕ over that of χ and taking the sum of the weights on its edges. Formally:

$$\chi \bullet \phi \triangleq \sum_{v \in V_G} \chi^{\mathsf{v}}(v, \phi^{\mathsf{v}}(v)) + \sum_{s \in n_G} \sum_{p \in \phi^{\mathsf{s}}(s)} \chi^{\mathsf{s}}(s,p) + \sum_{r \in m_G} \sum_{p \in \phi^{\mathsf{r}}(r)} \chi^{\mathsf{r}}(r,p) + \sum_{e \in E_G} \chi^{\mathsf{e}}(e, \phi^{\mathsf{e}}(e)) + \\ + \sum_{i \in Y_H^- \uplus X_H^+ \uplus P_H^+} \chi^{\mathsf{i}}(i, \phi^{\mathsf{i}}(i)) + \sum_{o \in Y_G^- \uplus X_G^+} \chi^{\mathsf{o}}(o, \phi^{\mathsf{o}}(o))$$

The assignment $\phi \mapsto \chi \bullet \phi$ defines a function from the set of embeddings $(G \hookrightarrow H)$ to $\mathbb{W}$.

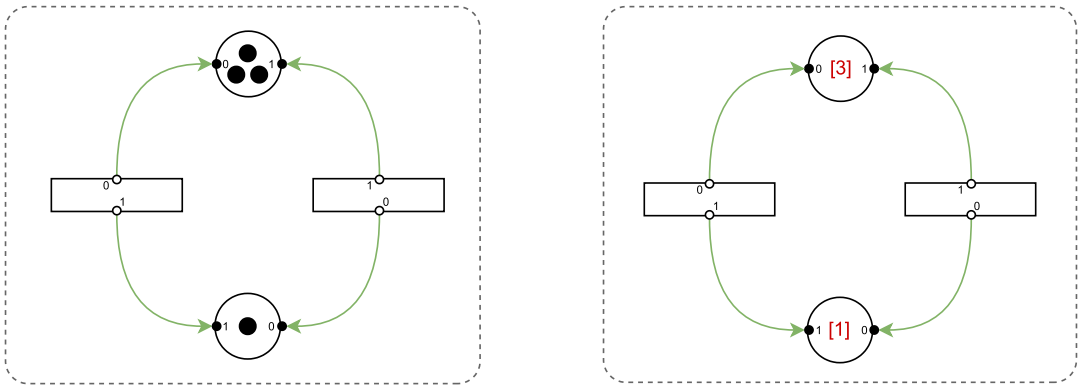


Fig. 14. A Petri net as directed graph (left) and as a weighted directed graph (right).

Weighted embeddings allow us to assign weights to different occurrences of the redex of a parametric reaction rule. As suggested above, we can use this information to weight reactions and refine a DBRS to its subsystem with optimal reactions. Another application is to extend the notion of DBRS from a nondeterministic to a quantitative semantics by replacing the reaction relation with a weighted one—we develop this direction in the next section.

Computing optimal embeddings Let $\mathcal{S}$ be the set of all assignments for the binary variables of DBGE$[G, H]$. Given an assignment $(\vec{N}, \vec{F}, \vec{M}) \in \mathcal{S}$, we can assign a weight $\chi \bullet (\vec{N}, \vec{F}, \vec{M})$ to it by reading it as a graph, laying it over the graph of χ and taking the sum of the weights on its edges. Formally:

$$\chi \bullet (\vec{N}, \vec{F}, \vec{M}) \triangleq \sum_{p,l} \chi(p, l) \cdot N_{p,l} + \sum_{h,h'} \chi(h, h') \cdot F_{h,h'} + \sum_{g,h} \chi(g, h) \cdot M_{g,h} \tag{39}$$

Proposition 22. *Let $\mathcal{S}$ be the set of all assignments for the binary variables of* DBGE$[G, H]$. *Any multi-linear map from $\mathcal{S}$ to $\mathbb{W}$ is induced by a $\mathbb{W}$-weighting for G and H.*

It follows that an instance of the optimal embedding problem can be solved using our algorithm whenever the cost function is induced by some $\mathbb{W}$-weighting.

The library jLibBig already offers experimental support for optimal weighted embeddings using Java integers as weights—other types can be easily added provided they are supported by the underlying solver. In practice, weights are defined as a function of data attached to the components of the guest and host bigraphs (called *attached properties*, in jLibBig).

7. Weighted bigraphs and reactive systems

In the previous section we weighted embeddings using functions that can be specific of the given guest and host bigraphs in an effort to keep the definition general. A less general but more practical approach is to weight the components of a graph and use this information to define weighing functions independently from the specific guest and host at hand. This consideration leads us to introduce *weighted directed bigraphs* and reactive systems over them.

7.1. Weighted directed bigraphs

Let $\mathbb{W}$ be a set of weights (additional structure will be added when needed).

Given a directed bigraph G, we call $\mathbb{W}$-weighting for G any function $\rho\colon |G| \to \mathbb{W}$ that assigns weights from $\mathbb{W}$ to the elements in the support of G. We define weighted versions of place, link, and bigraphs by equipping them with weighting.

Definition 23 *($\mathbb{W}$-Weighted Directed Bigraph).* A *W-weighted directed bigraph* is a pair $\langle G \cdot \rho \rangle$ where G is a directed graph and ρ is a $\mathbb{W}$-weighting for G.

We extend the graphical notation for directed bigraphs to the weighted case by labelling the components of their support with their weight; to avoid confusion with edge and node names we enclose weights in brackets. For instance, Fig. 14 contains two encodings of the same Petri net: in the first tokens are represented by individual nodes (as discussed in Example 8)) whereas in the second they are represented by weights attached to the nodes representing places. By representing tokens with weights instead of nodes we obtain a model that is more precise and more efficient. In fact, when tokens are

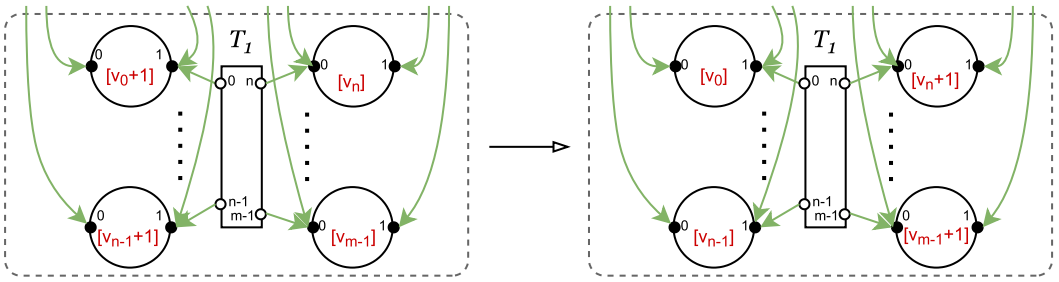


Fig. 15. Rewriting rule for a Petri net transition with n inputs and m outputs using weights to represent tokens.

represented by nodes they carry an identity even if they are all equivalent when it comes to the dynamics of Petri nets: a bigraph that represents exactly one token admits one embedding for each token in the host bigraph.

Here we have taken the design choice to weight only components in the support of bigraphs, and not those outside the support (e.g., links, names, interface widths). The reason is that these components may not be preserved by composition and juxtaposition; this makes unclear how to give them weights without committing to specific applications. For instance, assume links are also assigned weights; this can be done in a clean and elegant way by having the adjacency matrix for links take values in $\mathbb{W}$, and then composition can be defined in terms of matrix multiplication. Although the elegance of this solution is tempting, we lack a strong justification from the meta-modelling perspective. Bigraphs have been introduced as a meta-model for ubiquitous computing and as such any extension should prioritise expressiveness and generality.

The notion of support translation and support-equivalence (Definition 9) readily extends to weighted bigraphs: two weighted bigraphs $\langle G \cdot \rho \rangle$ and $\langle G' \cdot \rho' \rangle$ are support-equivalent if there is a support translation $\sigma : |G'| \to |G|$ such that $\sigma G = G'$ and $\rho \circ \sigma = \rho'$. We write $\sigma \langle G \cdot \rho \rangle$ for the weighted bigraph $\langle (\sigma G) \cdot (\rho \circ \sigma) \rangle$.

Given two $\mathbb{W}$-weightings $\rho_F : |F| \to \mathbb{W}$ and $\rho_G : |G| \to \mathbb{W}$ we write $(\rho_F \uplus \rho_G)$ for function given on any $x \in |F| \uplus |G|$ as follows:

$$(\rho_F \uplus \rho_G)(x) \triangleq \begin{cases} \rho_F(x) & \text{if } x \in |F| \\ \rho_G(x) & \text{if } x \in |G| \end{cases}$$

We define composition and juxtaposition by lifting of the corresponding operations for DBGs.

Definition 24 *(Composition and Juxtaposition).* For weighted directed bigraphs $\langle F \cdot \rho_F \rangle$ and $\langle G \cdot \rho_G \rangle$, their composition $\langle F \cdot \rho_F \rangle \circ \langle G \cdot \rho_G \rangle$ is the weighted directed bigraph $(G \circ F, \rho_G \uplus \rho_F)$. For weighted directed bigraphs $\langle F \cdot \rho_F \rangle$ and $\langle G \cdot \rho_G \rangle$, their juxtaposition $\langle F \cdot \rho_F \rangle \otimes \langle G \cdot \rho_G \rangle$ is the weighted directed bigraph $\langle G \otimes F \cdot \rho_G \uplus \rho_F \rangle$.

7.2. Reactive systems over weighted directed bigraphs

The first step for defining reactive systems over weighted directed bigraphs is to extend the notion of parametric reaction rule to act also on weightings.

Simply replacing bigraphs with weighted ones in Definition 13 is too restrictive: under this approach a parametric reaction rule is a triple $(\langle R : I \to J \cdot \rho_R \rangle, \langle R' : I' \to J \cdot \rho_{R'} \rangle, \eta : I \to I')$ and its application would require exact occurrence of the redex $\langle R : I \to J \cdot \rho_R \rangle$. This notion is too restrictive for many applications of interest. For instance, describing the dynamics of Petri nets would require an infinite number of rules, one for each possible marking. Instead, reaction rules should be parametrised (also) on weightings and describe how the weighting associated to the redex is replaced by a weighting for the reactum. To this end, we extend Definition 13 with a (possibly) partial function mapping weightings for the redex to weightings for the reactum. Partiality allows us to encode side conditions for the applicability of a rule (e.g., that a place must have at least n tokens).

Definition 25. A *parametric reaction rule* for weighted directed bigraphs is a tuple of the form $(R : I \to J, R' : I' \to J, \eta : I \to I', \theta : \mathbb{W}^{|R|} \rightharpoonup \mathbb{W}^{|R'|})$ where R is the parametric redex, R' the parametric reactum, η is an instantiation map, and θ is a (possibly partial) function mapping weightings for R to weightings for R'.

Example 26. To describe the dynamics of Petri nets in our model using weighted bigraphs, adapt the generic rewriting rule for a transition with n inputs and m outputs given in Fig. 5 by modelling markings using weights. Because nodes representing places do not contain other nodes we do not to use sites further simplifying the rule. The result is given in Fig. 15 where we represent θ using expressions with unknowns $v_0, \ldots, v_{m-1}$.

We can now define the key notion of reactive systems over weighted directed bigraphs as an extension of that for directed bigraphs (Definition 15). Let $Ag(\mathcal{K}, \mathbb{W})$ be the set of agents (weighted bigraphs with no inner names nor sites) over a signature $\mathcal{K}$ and domain of weights $\mathbb{W}$.

Definition 27. A *reactive system over weighted directed bigraphs* $WBRG(\mathcal{K}, \mathcal{R})$ is defined by a signature $\mathcal{K}$ and a set $\mathcal{R}$ of parametric reaction rules for DBGs weighted over $\mathbb{W}$. A reactive system $WBG(\mathcal{K}, \mathcal{R})$ induces a rewriting relation $\twoheadrightarrow \subseteq Ag(\mathcal{K}, \mathbb{W}) \times Ag(\mathcal{K}, \mathbb{W})$ according to the following rule:

$$\frac{\begin{array}{c}(R, R', \eta, \theta) \in \mathcal{R} \qquad \theta(\rho_R) = \rho_{R'} \\ \langle A \cdot \rho_A \rangle = \langle C \cdot \rho_C \rangle \circ \left(\sigma \langle R \cdot \otimes \rangle \langle Id_Z \cdot \rho_{Id_Z} \rangle\right) \circ \langle \omega \cdot \rho_\omega \rangle \circ \left(\langle D_0 \cdot \rho_{D_0} \rangle \otimes \ldots \otimes \langle D_{m-1} \cdot \rho_{D_{m-1}} \rangle\right) \\ \langle A' \cdot \rho_{A'} \rangle = \langle C \cdot \rho_C \rangle \circ \left(\sigma' \langle R' \cdot \rho_{R'} \rangle \otimes \langle Id_Z \cdot \rho_{Id_Z} \rangle\right) \circ \langle \omega' \cdot \rho_{\omega'} \rangle \circ \circ \left(\langle D_{\eta(0)} \cdot \rho_{D_{\eta(0)}} \rangle \otimes \ldots \otimes \langle D_{\eta(m-1)} \cdot \rho_{D_{\eta(m-1)}} \rangle\right)\end{array}}{\langle A \cdot \rho_A \rangle \twoheadrightarrow \langle A' \cdot \rho_{A'} \rangle} \tag{40}$$

where the support translations σ and σ' agree on $|R| \cap |R'|$, ω and ω' are wiring maps (cf. Definition 15).

The notion of reactive system over weighted DBGs, is a direct generalisation of the one for the non-weighted setting: if we remove weightings from (40) (or equivalently, if the weightings are only the trivial one, constantly equal to 0) we obtain (1), the corresponding rule in Definition 15.

7.3. Weighted reactive systems over directed bigraphs

In this section we extend BRS in a different direction by adding weights to their dynamics akin to weighted transition systems [28]. This allows us to encode various quantitative aspects such as stochastic rates, by suitably choosing the set of weights and its structure.

Fix a commutative monoid structure for the set of weights $\mathbb{W}$ and let $+$ and 0 denote its operation and unit, respectively. We extend parametric reaction rules with a weight to describe the contribution of the rule to a rewriting.

Definition 28. A *weighted parametric reaction rule* for directed bigraphs is a tuple of the form $(R\colon I \to J, R'\colon I' \to J, \eta\colon I \to I', w)$ where R is the parametric redex, R' the parametric reactum, η is an instantiation map, and $w \in \mathbb{W}$ is the weight associated to the rule.

The weight of a rewriting is given by the sum of the contributions of all applicable reactions.

Definition 29. A *weighted reactive system over directed bigraphs* $BWRS(\mathcal{K}, \mathcal{R})$ is defined by a signature $\mathcal{K}$ and a set $\mathcal{R}$ of weighted parametric reaction rules for DBGs. A weighted reactive system $WBRS(\mathcal{K}, \mathcal{R})$ induces a function $\delta_{\mathcal{R}}\colon Ag(\mathcal{K}) \times Ag(\mathcal{K}) \to \mathbb{W}$ that assigns weights to rewritings as follows:

$$\delta_{\mathcal{R}}(A, A') \triangleq \sum \left\{ w \;\middle|\; \begin{array}{l}(R, R', \eta, w) \in \mathcal{R} \\ A = C \circ (\sigma R \otimes Id_Z) \circ \omega \circ (D_0 \otimes \ldots \otimes D_{m-1}) \\ A' = C \circ (\sigma' R' \otimes Id_Z) \circ \omega' \circ (D_{\eta^P(0)} \otimes \ldots \otimes D_{\eta^P(m'-1)})\end{array} \right\} \tag{41}$$

We write $A \overset{w}{\twoheadrightarrow} A'$ for $\delta_{\mathcal{R}}(A, A') = w$.

If $\mathbb{W}$ is the commutative monoid $\mathbb{B} = (\{\texttt{true}, \texttt{false}\}, \vee, \texttt{false})$ of Boolean values equipped with logical disjunction, then a function weighting rewritings is equivalent to a relation allowing us to regard a weighted reactive system over DBGs as a reactive system over DBGs. In particular, (41) instantiates to

$$\delta_{\mathcal{R}}(A, A') \triangleq \bigvee \left\{ w \;\middle|\; \begin{array}{l}(R, R', \eta, w) \in \mathcal{R} \\ A = C \circ (\sigma R \otimes Id_Z) \circ \omega \circ (D_0 \otimes \ldots \otimes D_{m-1}) \\ A' = C \circ (\sigma' R' \otimes Id_Z) \circ \omega' \circ (D_{\eta^P(0)} \otimes \ldots \otimes D_{\eta^P(m'-1)})\end{array} \right\}$$

which is equivalent to (1) once we remove all parametric rewriting rules in $\mathcal{R}$ with weight `false`.

7.4. Weighted reactive systems over weighted directed bigraphs

In this section we combine both extensions to bigraphical reactive systems discussed in the previous sections. Let $\mathbb{W}$ be a commutative monoid.

Definition 30. A *weighted parametric reaction rule* for weighted directed bigraphs is a tuple of the form $(R\colon I \to J, R'\colon I' \to J, \eta\colon I \to I', \theta\colon \mathbb{W}^{|R|} \rightharpoonup \mathbb{W}^{|R'|} \times \mathbb{W})$ where R is the parametric redex, R' the parametric reactum, η is an instantiation map, and θ is a partial function mapping weightings for R to weightings for R' and weights for the rule.

Definition 31. A *weighted reactive system over weighted directed bigraphs* $WBWRS(\mathcal{K}, \mathcal{R})$ is defined by a signature $\mathcal{K}$ and a set $\mathcal{R}$ of weighted parametric reaction rules for DBGs weighted over $\mathbb{W}$. A weighted reactive system $WBWRS(\mathcal{K}, \mathcal{R})$ induces a function $\delta_{\mathcal{R}}\colon Ag(\mathcal{K}) \times Ag(\mathcal{K}) \to \mathbb{W}$ that assigns weights to rewritings as follows:

$$\delta_{\mathcal{R}}(\langle A \cdot \rho_A\rangle, \langle A' \cdot \rho_{A'}\rangle) \triangleq \sum \left\{ w \;\middle|\; \begin{array}{l} (R, R', \eta, \theta) \in \mathcal{R} \qquad \theta(\rho_R) = (\rho_{R'}, w) \\ \langle A \cdot \rho_A\rangle = \langle C \cdot \rho_C\rangle \circ \left(\sigma \langle R \cdot \rho_R\rangle \otimes \left\langle Id_Z \cdot \rho_{Id_Z}\right\rangle\right) \circ \langle \omega \cdot \rho_\omega\rangle \circ \\ \qquad \circ \left(\left\langle D_0 \cdot \rho_{D_0}\right\rangle \otimes \ldots \otimes \left\langle D_{m-1} \cdot \rho_{D_{m-1}}\right\rangle\right) \\ \left\langle A' \cdot \rho_{A'}\right\rangle = \langle C \cdot \rho_C\rangle \circ \left(\sigma' \left\langle R' \cdot \rho_{R'}\right\rangle \otimes \left\langle Id_Z \cdot \rho_{Id_Z}\right\rangle\right) \circ \left\langle \omega' \cdot \rho_{\omega'}\right\rangle \circ \\ \qquad \circ \left(\left\langle D_{\eta(0)} \cdot \rho_{D_{\eta(0)}}\right\rangle \otimes \ldots \otimes \left\langle D_{\eta(m-1)} \cdot \rho_{D_{\eta(m-1)}}\right\rangle\right) \end{array} \right\}$$

We write $\langle A \cdot \rho_A\rangle \overset{w}{\rightarrow} \left\langle A' \cdot \rho_{A'}\right\rangle$ for $\delta_{\mathcal{R}}(\langle A \cdot \rho_A\rangle, \left\langle A' \cdot \rho_{A'}\right\rangle) = w$.

8. Conclusions and future work

In this paper, we have presented a new version of *directed* bigraphs and bigraphical reactive systems, which subsume many previous versions (such as Milner's bigraphs). For this kind of bigraphs we have provided a sound and complete algorithm for solving the embedding problem, based on a constraint satisfaction problem. The resulting model is compact with the number of variables and linear constraints are polynomially bounded by the size of the guest and host bigraphs. We adapted the algorithm to compute optimal embeddings provided that the cost function meets some mild criteria related multi-linearity; this is the first algorithm for computing optimal embeddings of bigraphs. Moreover, we have introduced new quantitative versions of weighted bigraphs, of weighted reactive systems over bigraphs, and their combination.

The algorithm has been successfully integrated into jLibBig, an extensible library for manipulating bigraphical reactive systems. At the time of this writing there are no "official" (or "widely recognized") benchmarks, nor any other algorithms or available tools that solve the directed bigraph embedding problem, to compare with; therefore, to evaluate our implementation we considered a concise use case that covers most features typically occurring in modelling agent-based ubiquitous systems. The empirical evaluation on this case looks promising. As expected, the major contributors to the cost of computing a BRS are two: searching for a solution to the CSP and translating it into the structures used by jLibBig to represent an embedding. The first phase depends on the solver and could benefit from solver-specific optimisations of the model and heuristic. The second phase depends on the data structures used by jLibBig to represent bigraphs and their embeddings: bigraphs are immutable objects and embeddings include complete (and costly to build) decomposition of the host bigraph into a context, redex, and parameters as required by the notion of *bigraphical matching*. We think that the cost of the second phase can be substantially reduced by switching to mutable structures for bigraphs and decoupling the structures used for representing bigraphical embeddings and matchings.

The proposed approach offers great flexibility: it can be easily applied also to other extensions of bigraphs and directed bigraphs e.g. bigraphs with sharing [9] or local directed bigraphs [7]. An interesting direction for future work would be to extend the algorithm also to stochastic and probabilistic bigraphs [29]; this would offer useful modelling and verification tools for quantitative aspects, e.g. for systems biology [3,14]. Approximated embeddings are supported in jLibBig, but still as experimental feature. In fact, the theoretical foundations of this extension have not been fully investigated yet, suggesting another line of research.

CRediT authorship contribution statement

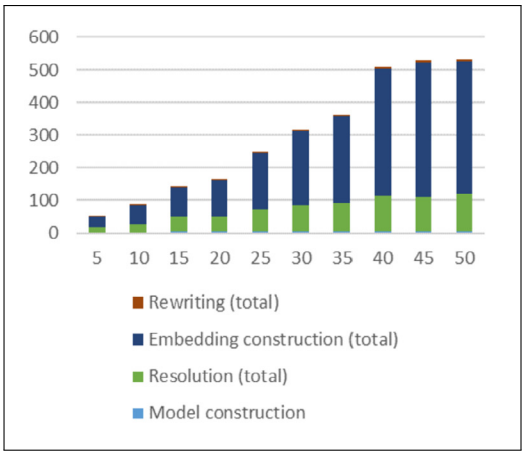
Alessio Chiapperini: Investigation, Software. **Marino Miculan:** Conceptualization, Formal analysis, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Marco Peressotti:** Conceptualization, Formal analysis, Software, Writing – original draft, Writing – review & editing.

Declaration of competing interest

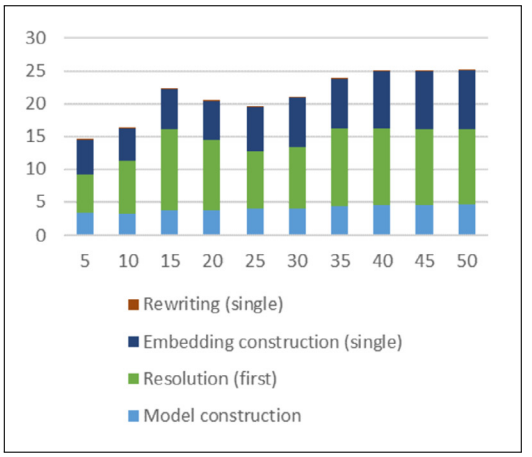
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Experimental results, omitted cases

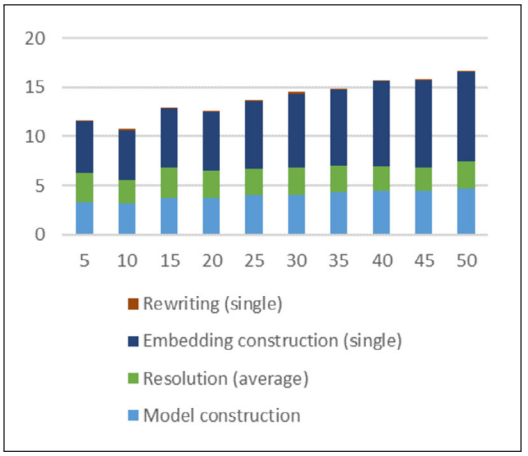
Experimental results for the cases omitted from Section 5 are reported in Figs. A.16–A.24.



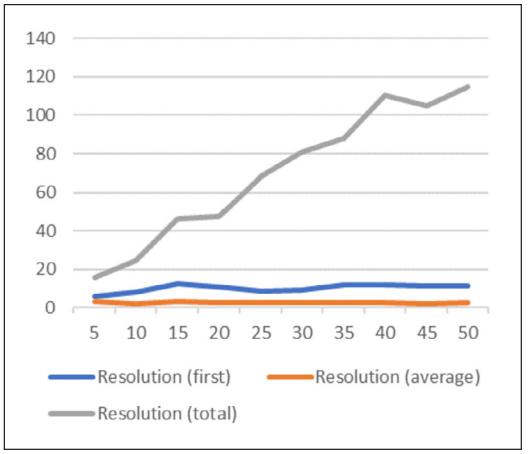
(a) First scenario, computing all reactions. Execution time (ms) vs. number of cars.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. number of cars.



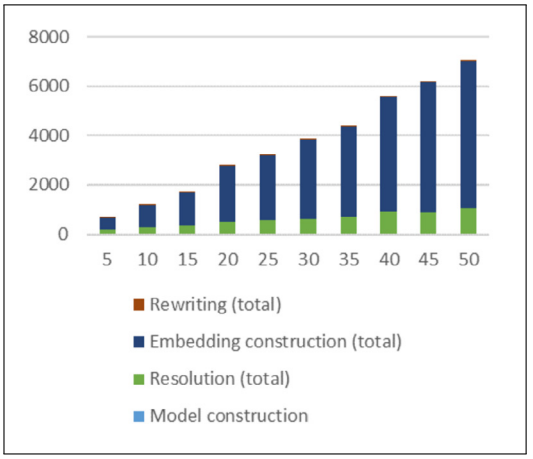
(c) Mean time for single reaction. Execution time (ms) vs. number of cars.



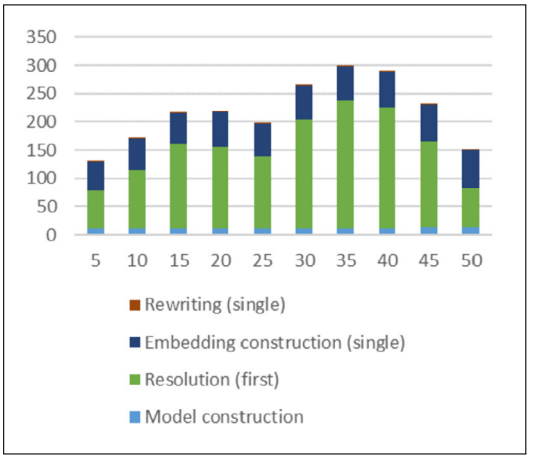
(d) Solution search. Execution time (ms) vs. number of cars.

Cars	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
5	6	3.3	5.8	3.0	15.9	5.3	31.2	0.086	0.5
10	11	3.2	8.0	2.4	24.8	5.1	56.5	0.112	1.2
15	15	3.7	12.4	3.2	46.4	6.0	90.6	0.081	1.2
20	18	3.7	10.7	2.8	47.7	6.0	110.7	0.078	1.4
25	25	4.0	8.7	2.8	68.5	6.8	173.2	0.103	2.6
30	30	4.0	9.3	2.8	80.9	7.6	229.5	0.117	3.5
35	35	4.4	11.8	2.7	88.0	7.7	265.7	0.110	3.8
40	44	4.5	11.7	2.5	110.4	8.7	390.1	0.136	6.0
45	45	4.5	11.5	2.3	105.1	8.9	412.8	0.119	5.4
50	44	4.7	11.3	2.7	114.5	9.1	407.9	0.087	3.8

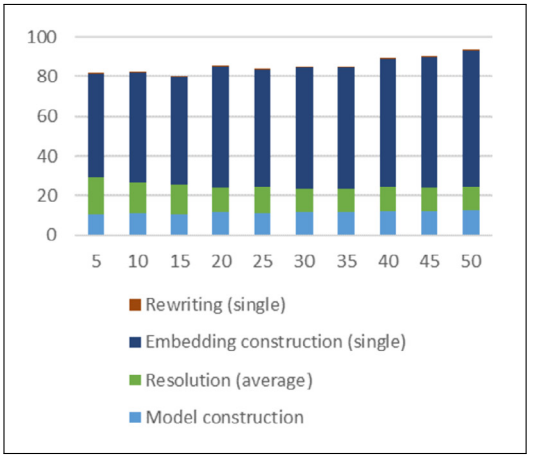
Fig. A.16. Execution time (ms) vs. number of cars, 5x5 grid with 50% connectivity (time values are in milliseconds).



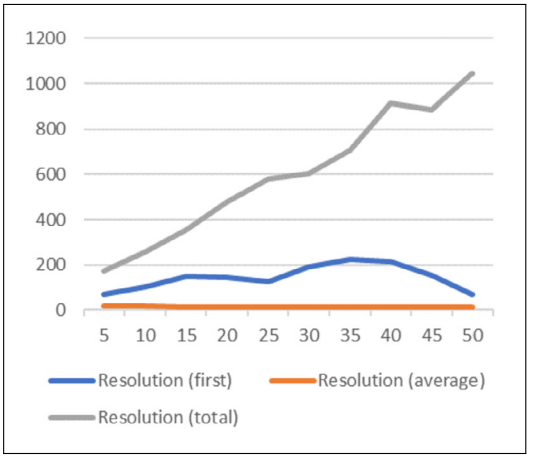
(a) First scenario, computing all reactions. Execution time (ms) vs. number of cars.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. number of cars.



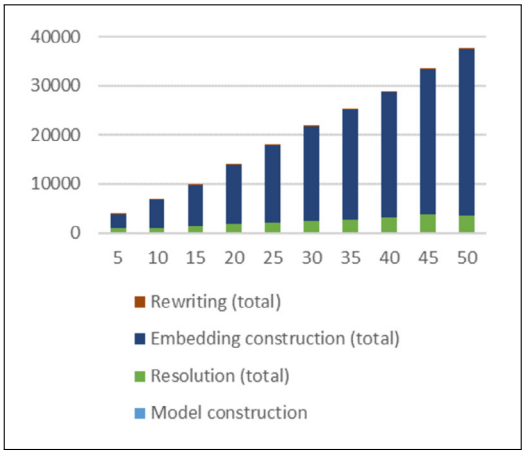
(c) Mean time for single reaction. Execution time (ms) vs. number of cars.



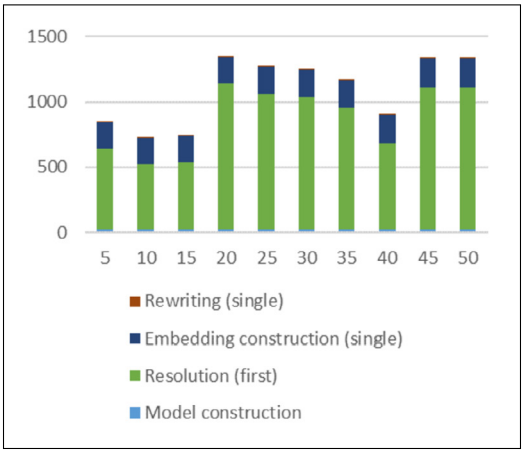
(d) Solution search. Execution time (ms) vs. number of cars.

Cars	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
5	10	10.5	67.8	18.9	171.6	52.0	496.8	0.240	2.3
10	16	10.9	104.0	16.0	258.5	55.1	912.4	0.244	4.0
15	25	10.8	150.2	14.5	357.0	54.6	1343.8	0.245	6.0
20	38	11.6	144.4	12.6	475.8	60.7	2294.3	0.212	8.0
25	44	11.3	127.0	13.3	581.0	59.1	2618.5	0.231	10.2
30	52	11.5	192.4	11.7	604.6	61.5	3203.8	0.222	11.5
35	59	11.4	226.3	11.9	709.2	61.3	3648.0	0.227	13.5
40	72	12.0	213.0	12.7	913.3	64.0	4609.0	0.244	17.5
45	79	12.3	152.7	11.4	886.7	66.1	5245.3	0.247	19.4
50	87	12.6	69.4	12.1	1046.3	68.3	5945.6	0.249	21.7

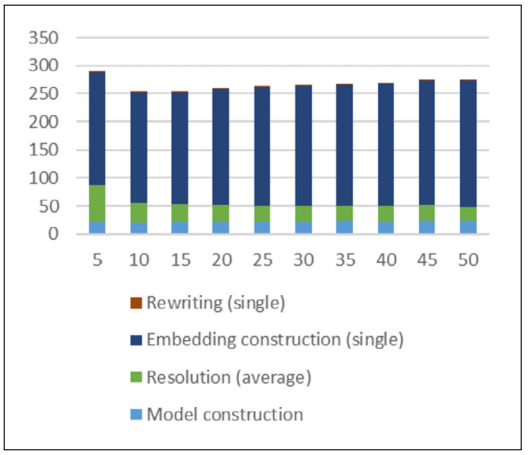
Fig. A.17. Execution time (ms) vs. number of cars, 8x8 grid with 70% connectivity (time values are in milliseconds).



(a) First scenario, computing all reactions. Execution time (ms) vs. number of cars.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. number of cars.

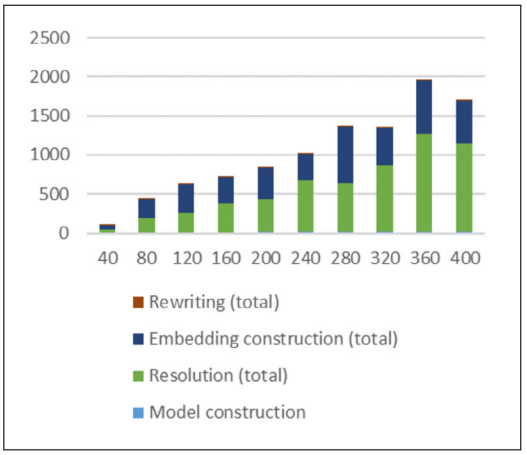


(c) Mean time for single reaction. Execution time (ms) vs. number of cars.

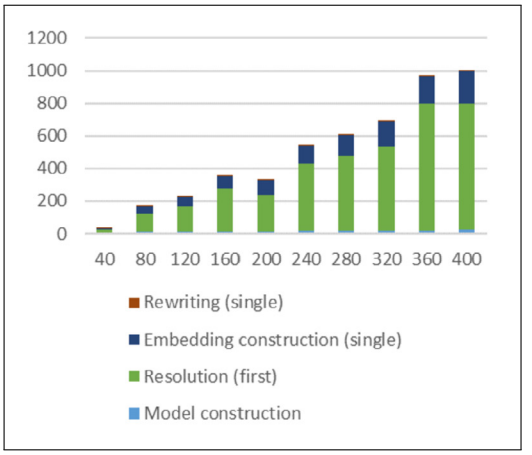
(d) Solution search. Execution time (ms) vs. number of cars.

Cars	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
5	14	21.1	621.1	66.9	864.6	199.6	2817.9	0.298	4.2
10	29	20.9	502.5	35.7	1031.8	196.2	5723.6	0.357	10.4
15	42	21.4	516.3	32.9	1386.1	199.0	8458.9	0.330	14.0
20	59	21.5	1116.2	31.2	1816.8	205.3	12051.6	0.329	19.3
25	74	22.2	1035.2	27.9	2067.9	211.8	15754.5	0.314	23.3
30	90	22.2	1011.8	27.9	2505.0	213.6	19188.0	0.349	31.3
35	104	23.4	930.4	26.4	2761.0	215.3	22478.9	0.364	38.0
40	117	22.8	659.4	26.7	3120.8	218.5	25473.4	0.348	40.6
45	133	23.4	1085.6	27.7	3687.4	222.1	29607.4	0.355	47.3
50	149	23.9	1085.1	24.3	3610.7	225.8	33696.7	0.347	51.7

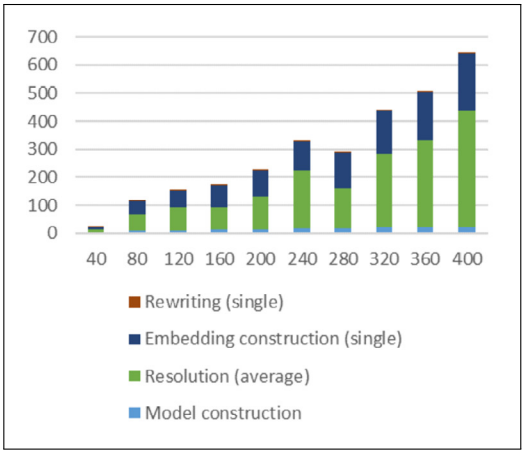
Fig. A.18. Execution time (ms) vs. number of cars, 10x10 grid with 90% connectivity (time values are in milliseconds).



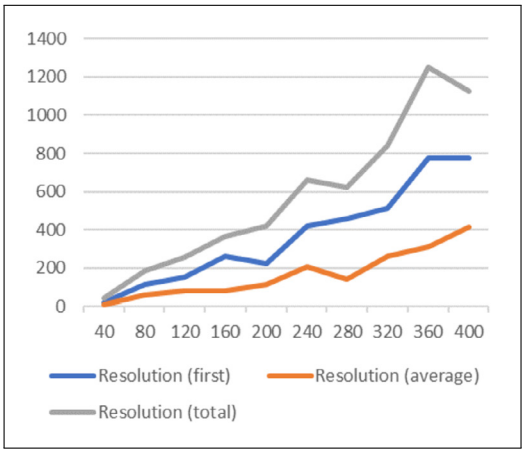
(a) First scenario, computing all reactions. Execution time (ms) vs. grid size.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. grid size.



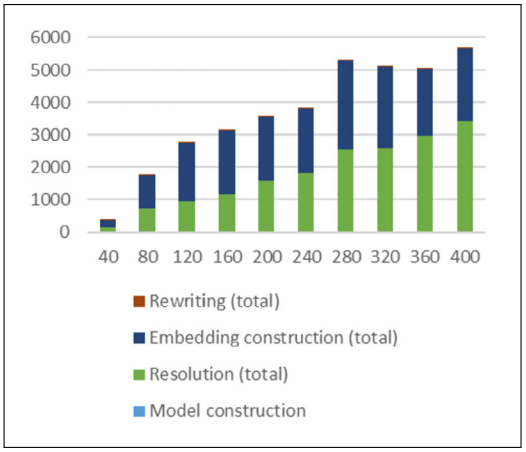
(c) Mean time for single reaction. Execution time (ms) vs. grid size.



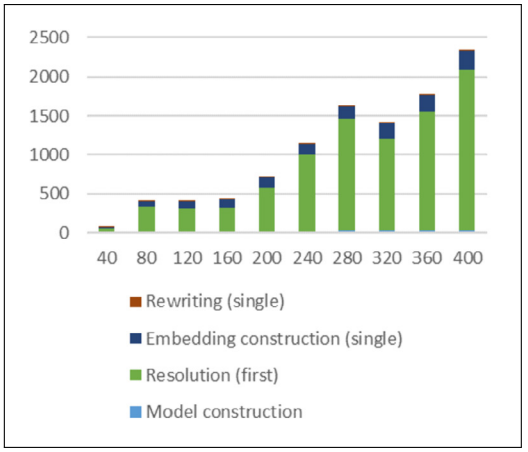
(d) Solution search. Execution time (ms) vs. grid size.

Zones	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
40	6	4.0	22.2	9.4	44.4	8.7	51.1	0.050	0.3
80	5	10.3	112.6	57.1	183.2	47.1	244.8	0.294	1.5
120	6	11.8	155.1	81.3	255.6	60.8	351.9	0.190	1.1
160	5	13.1	264.1	81.6	367.0	75.5	339.7	0.333	1.5
200	4	14.8	221.5	116.6	421.4	91.5	396.9	0.341	1.5
240	3	16.2	417.8	207.2	663.1	105.2	336.7	0.281	0.9
280	6	17.9	458.9	141.8	621.2	127.0	727.9	0.421	2.4
320	3	20.0	515.8	263.0	841.7	153.5	491.1	0.438	1.4
360	4	21.3	773.2	312.6	1250.5	170.8	683.0	0.350	1.4
400	3	23.0	773.3	416.4	1124.4	201.6	544.2	0.370	1.0

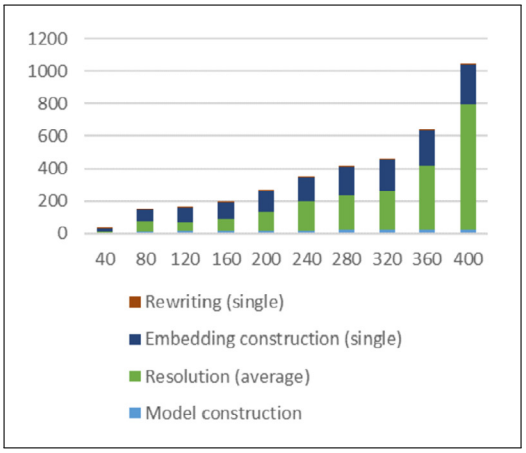
Fig. A.19. Execution time (ms) vs. grid size, 10 cars and 55% connectivity (values are in milliseconds).



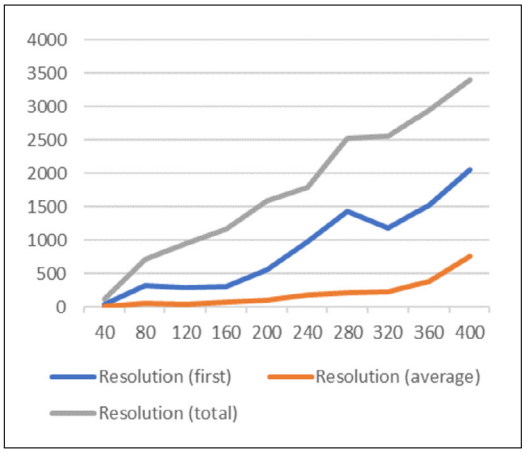
(a) First scenario, computing all reactions. Execution time (ms) vs. grid size.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. grid size.



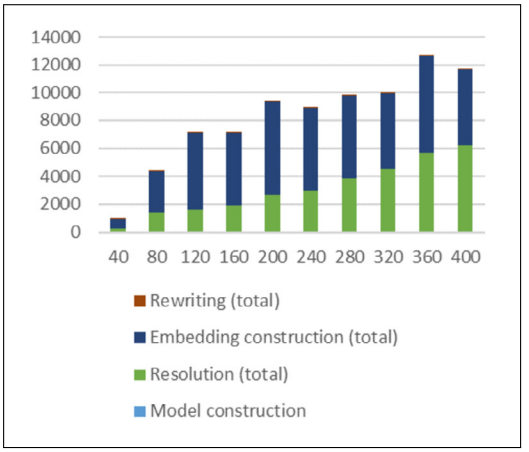
(c) Mean time for single reaction. Execution time (ms) vs. grid size.



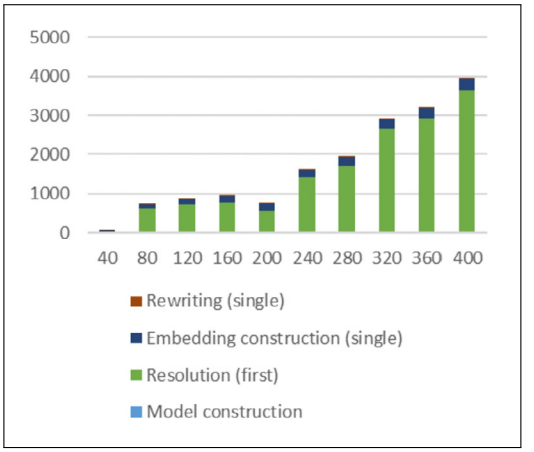
(d) Solution search. Execution time (ms) vs. grid size.

Zones	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
40	17	5.5	44.5	8.3	125.9	14.3	241.9	0.103	1.7
80	14	12.4	318.8	60.5	711.2	72.1	1022.3	0.241	3.4
120	20	14.5	294.2	51.7	945.8	90.7	1775.5	0.230	4.5
160	18	16.0	310.4	69.6	1169.3	106.8	1949.9	0.322	5.9
200	15	17.7	555.3	114.0	1591.2	130.4	1958.5	0.360	5.4
240	14	19.3	979.7	181.3	1796.6	146.2	1995.4	0.331	4.5
280	16	20.8	1435.9	217.3	2534.1	170.8	2733.9	0.340	5.4
320	13	22.5	1183.1	236.6	2554.4	196.8	2534.1	0.391	5.0
360	10	24.0	1530.3	390.7	2944.5	218.9	2072.5	0.295	2.8
400	9	25.8	2058.9	769.8	3408.6	247.3	2208.5	0.478	4.3

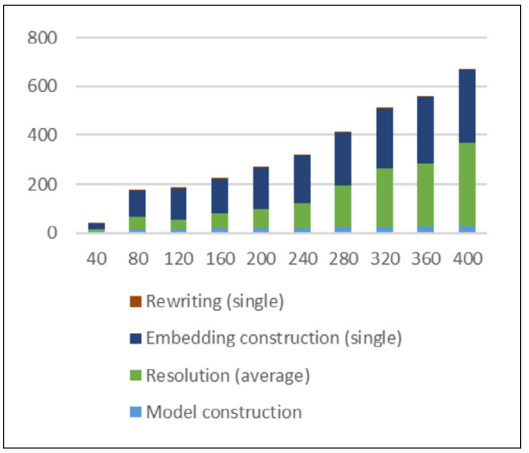
Fig. A.20. Execution time (ms) vs. grid size, 30 cars and 70% connectivity (values are in milliseconds).



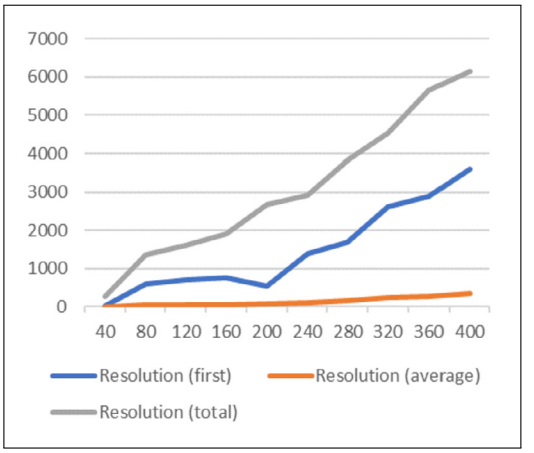
(a) First scenario, computing all reactions. Execution time (ms) vs. grid size.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. grid size.



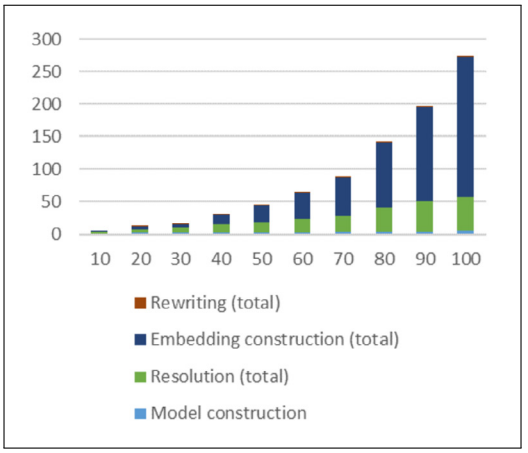
(c) Mean time for single reaction. Execution time (ms) vs. grid size.



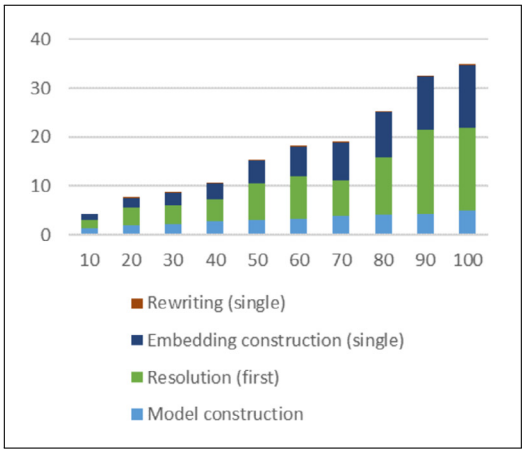
(d) Solution search. Execution time (ms) vs. grid size.

Zones	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
40	30	7.0	28.9	9.9	274.2	21.7	662.6	0.164	5.0
80	29	15.8	607.5	52.6	1355.1	106.6	3048.0	0.276	7.9
120	43	17.7	702.9	38.5	1611.0	126.8	5473.6	0.306	13.2
160	36	19.0	764.4	61.2	1914.4	143.1	5140.3	0.298	10.7
200	39	20.7	545.3	76.6	2665.1	171.7	6686.1	0.324	12.6
240	31	22.2	1394.2	102.3	2923.0	190.1	5983.3	0.385	12.1
280	27	23.6	1701.1	171.0	3834.7	216.2	5904.7	0.402	10.9
320	22	26.0	2631.0	240.8	4540.7	241.8	5409.3	0.441	9.8
360	26	27.8	2896.9	260.1	5656.1	270.2	6930.8	0.389	10.0
400	19	28.8	3607.1	340.6	6141.8	298.3	5515.8	0.405	7.5

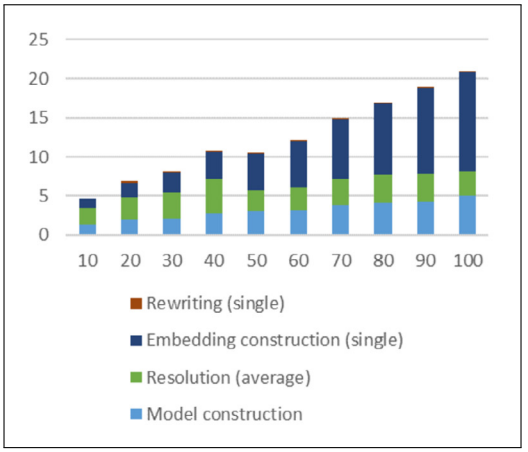
Fig. A.21. Execution time (ms) vs. grid size, 50 cars and 85% connectivity (values are in milliseconds).



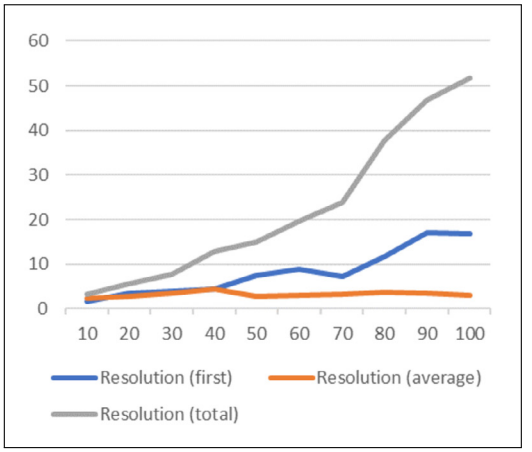
(a) First scenario, computing all reactions. Execution time (ms) vs. connectivity.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. connectivity.



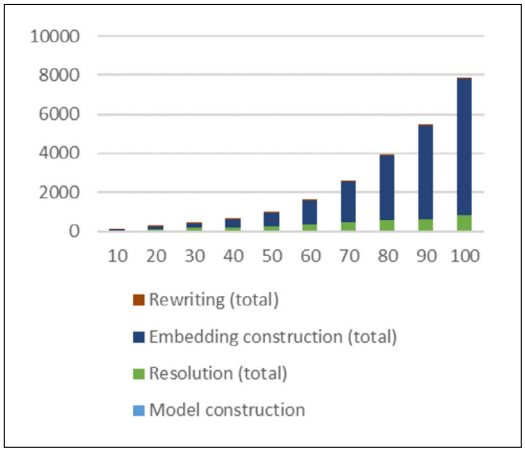
(c) Mean time for single reaction. Execution time (ms) vs. connectivity.



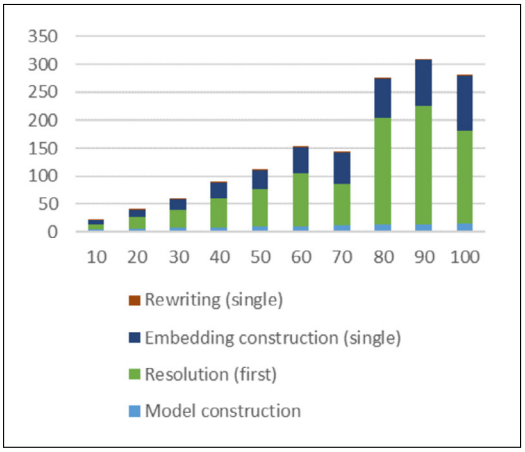
(d) Solution search. Execution time (ms) vs. connectivity.

Connectivity	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
10	2	1.3	1.7	2.2	3.3	1.2	1.8	0.000	0.0
20	2	2.0	3.6	2.8	5.6	1.9	3.8	0.200	0.4
30	2	2.1	4.0	3.4	7.8	2.6	5.9	0.043	0.1
40	4	2.8	4.4	4.4	12.8	3.4	14.6	0.049	0.2
50	5	3.0	7.4	2.7	14.9	4.7	26.1	0.167	0.9
60	7	3.2	8.9	3.0	19.6	5.8	40.1	0.043	0.3
70	8	3.9	7.3	3.4	23.8	7.6	59.9	0.104	0.8
80	11	4.1	11.7	3.6	37.6	9.2	99.5	0.103	1.1
90	13	4.3	17.1	3.6	46.9	10.9	144.0	0.098	1.3
100	17	5.0	16.9	3.1	51.8	12.8	215.1	0.119	2.0

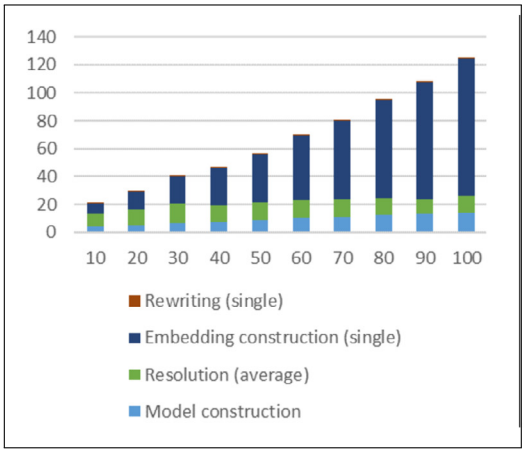
Fig. A.22. Execution time (ms) vs. connectivity, 5x5 grid and 5 cars (values are in milliseconds).



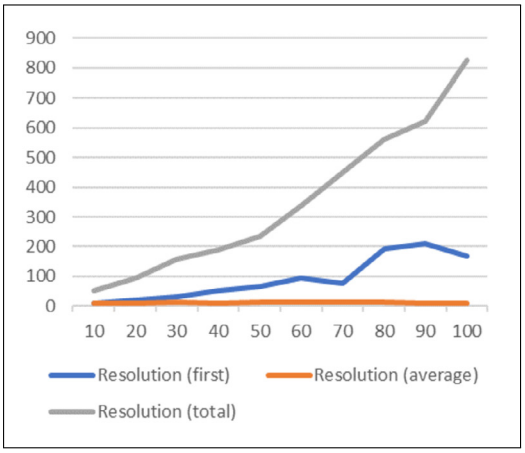
(a) First scenario, computing all reactions. Execution time (ms) vs. connectivity.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. connectivity.



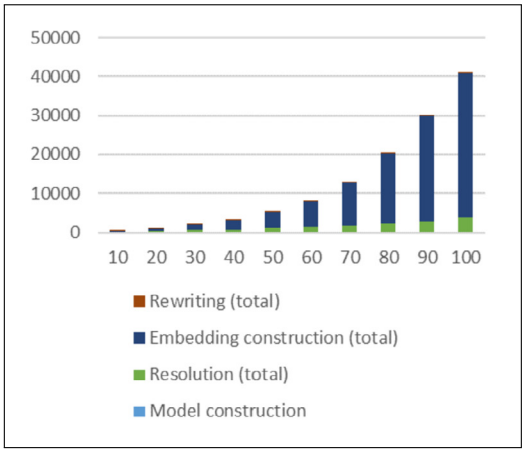
(c) Mean time for single reaction. Execution time (ms) vs. connectivity.



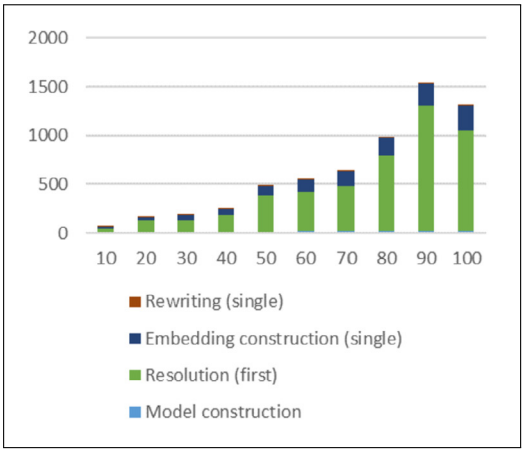
(d) Solution search. Execution time (ms) vs. connectivity.

Connectivity	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
10	6	4.0	9.4	9.6	50.6	7.1	43.6	0.066	0.4
20	10	5.3	21.7	10.9	95.0	12.6	123.6	0.113	1.1
30	12	6.7	32.7	13.9	157.8	19.5	237.4	0.157	1.9
40	17	7.5	53.4	11.6	189.7	26.8	442.5	0.182	3.0
50	20	8.6	67.4	12.6	234.7	35.2	692.9	0.190	3.7
60	26	10.1	95.6	13.0	336.9	46.8	1247.1	0.186	4.9
70	36	11.0	75.9	12.5	448.6	56.6	2066.7	0.245	8.9
80	47	12.4	191.3	12.0	560.4	70.6	3307.8	0.235	11.0
90	57	13.1	211.3	11.0	622.8	84.0	4813.2	0.253	14.5
100	71	14.2	167.3	11.7	824.9	98.8	6975.2	0.272	19.2

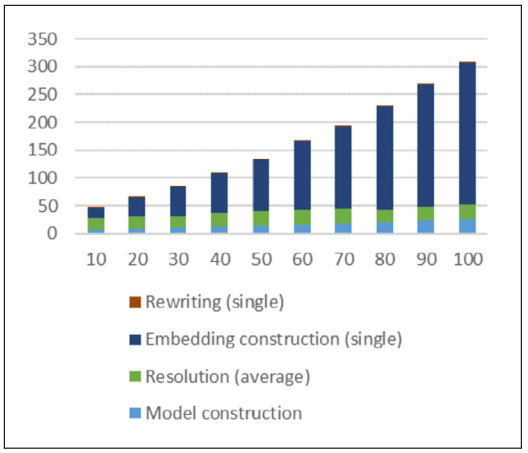
Fig. A.23. Execution time (ms) vs. connectivity, 8x8 grid and 20 cars (values are in milliseconds).



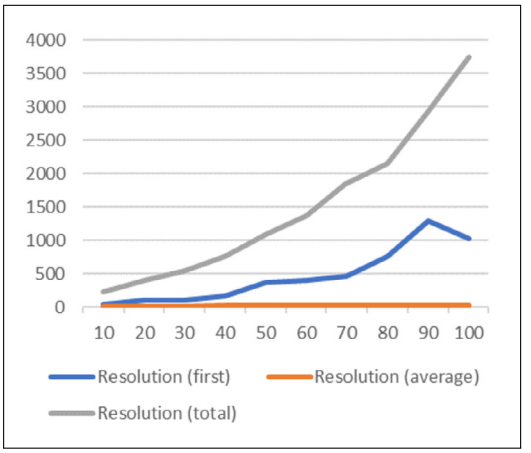
(a) First scenario, computing all reactions. Execution time (ms) vs. connectivity.



(b) Second scenario, computing the first reaction. Execution time (ms) vs. connectivity.



(c) Mean time for single reaction. Execution time (ms) vs. connectivity.



(d) Solution search. Execution time (ms) vs. connectivity.

Connectivity	Solutions	Model (ms)	Solution Search (ms)			Embedding (ms)		Rewriting (ms)	
			First	Average	Total	Single	Total	Single	Total
10	12	6.8	40.8	20.4	226.2	19.5	230.8	0.121	1.4
20	19	9.2	113.9	21.6	399.6	34.0	661.7	0.145	2.8
30	27	11.1	114.7	21.1	547.7	52.5	1419.4	0.208	5.6
40	32	13.1	167.8	24.6	761.9	70.0	2221.1	0.212	6.7
50	44	15.0	374.0	25.5	1087.4	92.5	4054.4	0.291	12.7
60	54	17.2	404.1	26.2	1368.9	122.5	6617.4	0.304	16.3
70	73	18.7	464.5	25.4	1853.9	149.1	10962.9	0.332	24.4
80	97	20.8	770.9	22.4	2152.5	184.9	17944.8	0.302	29.3
90	122	23.2	1288.6	24.2	2931.3	220.5	26835.5	0.321	39.0
100	145	25.7	1027.4	25.8	3741.7	255.8	37083.4	0.345	50.0

Fig. A.24. Execution time (ms) vs. connectivity, 10x10 grid and 40 cars (values are in milliseconds).

References

[1] B. Archibald, K. Burns, C. McCreesh, M. Sevegnani, Practical bigraphs via subgraph isomorphism, in: L.D. Michel (Ed.), 27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021, in: LIPIcs, vol. 210, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 15:1–15:17.
[2] B. Archibald, M.-Z. Shieh, Y.-H. Hu, M. Sevegnani, Y.-B. Lin Bigraphtalk, Verified design of iot applications, IEEE Int. Things J. 7 (4) (2020) 2955–2967.
[3] G. Bacci, D. Grohmann, M. Miculan, Bigraphical models for protein and membrane interactions, in: G. Ciobanu (Ed.), Proc. MeCBIC 2009, in: EPTCS, vol. 11, 2009, pp. 3–18.
[4] G. Bacci, D. Grohmann, M. Miculan, DBtk: a toolkit for directed bigraphs, in: Proc. CALCO, Springer, 2009, pp. 413–422.
[5] G. Bacci, M. Miculan, R. Rizzi, Finding a forest in a tree, in: International Symposium on Trustworthy Global Computing, Springer, 2014, pp. 17–33.
[6] M. Bundgaard, A.J. Glenstrup, T. Hildebrandt, E. Højsgaard, H. Niss, Formalizing higher-order mobile embedded business processes with binding bigraphs, in: Proc. COORDINATION, Springer, 2008, pp. 83–99.
[7] F. Burco, M. Miculan, M. Peressotti, Towards a formal model for composable container systems, in: Proc. SAC, ACM, 2020, pp. 173–175.
[8] M. Calder, A. Koliousis, M. Sevegnani, J.S. Sventek, Real-time verification of wireless home networks using bigraphs with sharing, Sci. Comput. Program. 80 (2014) 288–310.
[9] M. Calder, M. Sevegnani, Process algebra for event-driven runtime verification: a case study of wireless network management, in: Int. Conference on Integrated Formal Methods, Springer, 2012, pp. 21–23.
[10] A. Chiapperini, M. Miculan, M. Peressotti, Computing embeddings of directed bigraphs, in: F. Gadducci, T. Kehrer (Eds.), Proc. ICGT, in: LNCS, vol. 12150, Springer, 2020, pp. 38–56.
[11] A. Chiapperini, M. Miculan, M. Peressotti, A CSP implementation of the directed bigraph embedding problem, CoRR, arXiv:2003.10209 [abs], 2020.
[12] A. Chiapperini, M. Miculan, M. Peressotti, An evaluation of jLibBig rewriting engine for directed bigraphs, Available at https://doi.org/10.5281/zenodo.6546652, May 2022.
[13] T.C. Damgaard, A.J. Glenstrup, L. Birkedal, R. Milner, An inductive characterization of matching in binding bigraphs, Form. Asp. Comput. 25 (2) (2013) 257–288.
[14] T.C. Damgaard, E. Højsgaard, J. Krivine, Formal cellular machinery, Electron. Notes Theor. Comput. Sci. 284 (2012) 55–74.
[15] S. Debois, Sortings and bigraphs, Ph.d. Thesis, IT University of Copenhagen, 2008.
[16] A.J. Faithfull, G. Perrone, T. Hildebrandt, Big red: a development environment for bigraphs, Electron. Commun. EASST 61 (2013).
[17] A. Gassara, I.B. Rodriguez, M. Jmaiel, K. Drira, Executing bigraphical reactive systems, Discrete Appl. Math. 253 (2019) 73–92.
[18] A.J. Glenstrup, T.C. Damgaard, L. Birkedal, E. Højsgaard, An implementation of bigraph matching, IT University of Copenhagen, 2007, p. 22.
[19] D. Grohmann, Security, cryptography and directed bigraphs, in: Proc. ICGT, in: LNCS, vol. 5214, Springer, 2008, pp. 487–489.
[20] D. Grohmann, M. Miculan, Directed bigraphs, Electron. Notes Theor. Comput. Sci. 173 (2007) 121–137.
[21] D. Grohmann, M. Miculan, Reactive systems over directed bigraphs, in: Proc. CONCUR, in: LNCS, vol. 4703, Springer, 2007, pp. 380–394.
[22] D. Grohmann, M. Miculan, Controlling resource access in directed bigraphs, Electron. Commun. EASST 10 (2008).
[23] D. Grzelak, Bigraph framework: a framework written in Java for the manipulation and simulation of bigraphical reactive systems, Available at https://bigraphs.org/products/bigraph-framework/, Jan. 2022.
[24] E. Højsgaard, Bigraphical languages and their simulation, PhD thesis, IT University of Copenhagen, 2012.
[25] E. Højsgaard, A.J. Glenstrup, The bpl tool: a tool for experimenting with bigraphical reactive systems, in: Bigraphical Languages and Their Simulation, 2011, p. 85.
[26] O.H. Jensen, R. Milner, Bigraphs and Transitions, SIGPLAN Notices, vol. 38, ACM, 2003, pp. 38–49.
[27] S. Khanna, M. Sudan, L. Trevisan, D.P. Williamson, The approximability of constraint satisfaction problems, SIAM J. Comput. 30 (6) (2001) 1863–1920.
[28] B. Klin, V. Sassone, Structural operational semantics for stochastic and weighted transition systems, Inf. Comput. 227 (2013) 58–83.
[29] J. Krivine, R. Milner, A. Troina, Stochastic bigraphs, Electron. Notes Theor. Comput. Sci. 218 (2008) 73–96.
[30] A. Mansutti, M. Miculan, M. Peressotti, Distributed execution of bigraphical reactive systems, Electron. Commun. EASST 71 (2014).
[31] A. Mansutti, M. Miculan, M. Peressotti, Multi-agent systems design and prototyping with bigraphical reactive systems, in: K. Magoutis, P. Pietzuch (Eds.), Proc. DAIS, in: LNCS, vol. 8460, Springer, 2014, pp. 201–208.
[32] A. Mansutti, M. Miculan, M. Peressotti, Towards distributed bigraphical reactive systems, in: R. Echahed, A. Habel, M. Mosbah (Eds.), Proc. GCM'14, 2014, p. 45.
[33] M. Miculan, M. Peressotti, Bigraphs reloaded: a presheaf presentation, Technical Report UDMI/01/2013, Dept. of Mathematics and Computer Science, Univ. of Udine, 2013.
[34] M. Miculan, M. Peressotti, A CSP implementation of the bigraph embedding problem, CoRR, arXiv:1412.1042 [abs], 2014.
[35] M. Miculan, M. Peressotti, jLibBig: a library for bigraphical reactive systems, Available at https://bigraphs.github.io/jlibbig/, Nov. 2015.
[36] R. Milner, The Space and Motion of Communicating Agents, Cambridge University Press, 2009.
[37] J. Parrow, B. Victor, The fusion calculus: expressiveness and symmetry in mobile processes, in: Proc. LICS, IEEE, 1998, pp. 176–185.
[38] G. Perrone, S. Debois, T. Hildebrandt, Bigraphical refinement, in: Refine@FM, in: EPTCS, vol. 55, 2011, pp. 20–36.
[39] G. Perrone, S. Debois, T. Hildebrandt, A model checker for bigraphs, in: Proc. SAC, 2012, pp. 1320–1325.
[40] C. Prud'homme, J.-G. Fages, X. Lorca, Choco Documentation, TASC - LS2N CNRS UMR 6241, COSLING S.A.S., 2017.
[41] H. Sahli, T. Ledoux, É. Rutten, Modeling self-adaptive fog systems using bigraphs, in: Proc. FOCLASA, 2019, pp. 1–16.
[42] V. Sassone, P. Sobocinski, Reactive systems over cospans, in: Proc. 20th Symposium on Logic in Computer Science, LICS 2005, IEEE, 2005, pp. 311–320.
[43] M. Sevegnani, M. Calder, Bigraphs with sharing, Theor. Comput. Sci. 577 (2015) 43–73.
[44] M. Sevegnani, M. Calder, BigraphER: rewriting and analysis engine for bigraphs, in: Computer Aided Verification - 28th International Conference, CAV 2016, 2016, pp. 494–501.
[45] M. Sevegnani, M. Calder, Bigrapher: rewriting and analysis engine for bigraphs, in: S. Chaudhuri, A. Farzan (Eds.), Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II, in: Lecture Notes in Computer Science, vol. 9780, Springer, 2016, pp. 494–501.
[46] M. Sevegnani, C. Unsworth, M. Calder, A SAT based algorithm for the matching problem in bigraphs with sharing, Tech. Rep., University of Glasgow, 2010.
[47] M. Souad, B. Faiza, H. Nabil, Formal modeling iot systems on the basis of biagents* and maude, in: International Conference on Advanced Aspects of Software Engineering (ICAASE), 2020, pp. 1–7.
[48] C. Tsigkanos, T. Kehrer, C. Ghezzi, Modeling and verification of evolving cyber-physical spaces, in: E. Bodden, W. Schäfer, A. van Deursen, A. Zisman (Eds.), Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4–8, 2017, ACM, 2017, pp. 38–48.
[49] C. Tsigkanos, N. Li, Z. Jin, Z. Hu, C. Ghezzi, Scalable multiple-view analysis of reactive systems via bidirectional model transformations, in: 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21–25, 2020, IEEE, 2020, pp. 993–1003.
[50] C. Tsigkanos, L. Pasquale, C. Ghezzi, B. Nuseibeh, On the interplay between cyber and physical spaces for adaptive security, IEEE Trans. Dependable Secure Comput. 15 (3) (2018) 466–480.