

From Goals to Organisations: automated organisation generator for MAS*

Cleber Jorge Amaral^{1,2}[0000-0003-3877-6114] and
Jomi Fred Hübner²[0000-0001-9355-822X]

¹ Federal Institute of Santa Catarina (IFSC), São José, SC, Brazil
`cleber.amaral@ifsc.edu.br`

<http://www.ifsc.edu.br/>

² Federal University of Santa Catarina (UFSC), Florianópolis, SC, Brazil
`jomi.hubner@ufsc.br`
<http://pgeas.ufsc.br/en/>

Abstract. An explicit organisational structure helps entrants in open multi-agent systems (MAS) to reason about their positions in the organisation for cooperating to achieve mutual goals. In spite of its importance, there are few studies on automatic organisation generators that create explicit organisational structures. This paper introduces *GoOrg*, a proposal for automatic design of organisations. Our approach considers as inputs a goal decomposition tree (*gdt*) and user preferences. From the *gdt* with annotations such as necessary skills to achieve organisational goals, predicted workload and throughput, *GoOrg* creates roles in the form of an organisational chart. The main challenge is to define strategies to search the space of all organisational structures for those that can achieve the goals respecting constraints and taking into account user preferences. We can, for instance, prefer a *flatter* or a *taller* structure, more specialist or more *generalist* roles, and we can accept *matrix connections* or not.

Keywords: Automated organisation design · Organisational chart · Organisational structure · Open Multi-Agent Systems

1 Introduction

The organisational structure is an instrument used to split, organise and coordinate activities of Multi-Agent System (MAS) organisations. It reflects authority relations and responsibility for goals, providing a typical way to assign tasks to agents [15]. An explicit organisational structure helps agents to know where they fit relatively to others and which are their responsibilities [9, 13, 21, 30].

Currently, there are a few studies on the automatic design of organisations that generates explicit organisational structures [23, 10, 26, 17]. Although seminal, there is still space for improvements, for instance, automating the roles creation process. This paper presents an ongoing work in the context of a PhD

* Supported by Petrobras project AG-BR, IFSC and UFSC.

thesis that proposes *GoOrg*, an automated organisation generator which takes a goal decomposition tree (*gdt*) and produces as output an organisational chart, i.e., an explicit organisational structure, according to user preferences. The main novelty of our method is its capability of creating roles from the inputs. In this sense, our method may produce a larger range of possible organisational charts.

To discuss the problem and to describe the proposed generator, Section 2 presents the concept of automatic organisation design and the state of art of this research area. Section 3 describes the problem, i.e., the challenge we want to overcome. Section 4 presents our organisation generator *GoOrg*. Section 5 describes the research method we are applying in, the status of this research and planned evaluation. Finally, Section 6 presents related works and Section 7 presents our conclusions.

2 Organisation design

Pattinson et al [22] define organisation design as “the problem of choosing the best organisation class - from a set of class descriptions - given knowledge about the organisation’s purpose (goal, task, and constraints on the goal) and the environment in which the organisation is to operate”. Given necessary input, an organisation generator can give as output organisational aspects, such as, structure, goals definitions, strategy, how leadership will work, which reward system will be used, among others [2]. We have identified three classes of organisation generators in the MAS domain [1].

The first class is the *automated organisation design by task planning*. These generators usually create *problem-driven* organisations, for specific and generally short term purposes [8]. The organisational structure is typically not explicit being an unintended result of a task allocation process. Such generators are focused on solving a given problem by decomposing tasks, allocating them on the available agents [4]. Agents are previously known, and usually, roles are not necessary. The agents generally cooperate by fulfilling their tasks which, when combined, implies in the achievement of global goals.

The second class uses *self-organisation* approaches. In this class, the organisations usually emerge by agents common interest and interactions [13]. Resulting organisations are dynamic, may operate continuously, have overlapping tasks, have no external or central control, and hierarchy and information flow in many directions [32]. The organisational structure is an informal implicit outcome of this bottom-up process. The target of this method is to solve some problem and not precisely to carefully design an organisation [26, 28].

Finally, the third class is the *automated explicit organisation generators*. It is focused “on a specification of desired outcomes and the course of actions for achieving them, analysis of the organisational environment and available resources, allocation of those resources and development of organisational structures and control system” [15]. It considers inputs such as organisational goals, available agents, resources and performance targets, producing explicit organi-

sation definitions, which may include roles, constraints, assignments of responsibilities, hierarchy and other relations.

The first class can provide a very efficient way to allocate tasks among agents when the MAS is solving a previously known problem, usually in deterministic environments. However, it may lack the ability to deal with entrants in case of open systems, because it is supposed to know at planning time the available agents. In this sense, a new agent would not know what to do and how to cooperate unless a *replanning* is triggered, which can be computationally heavy.

Whether dealing with uncertainty and dynamic environments, the second class has advantages over other classes, which cannot deal with unpredictable situations [13]. However, in some cases, an entrant of an open system would need to negotiate with other agents his participation what may be slow to accommodate due to message exchanging.

Alternatively, the latter class cares on designing explicit structures which foster entrances and exits [11]. When adopting a role, an entrant receives its responsibilities, starting to cooperate with other organisational members. In many cases, an entrance does not require any extra designing effort since the roles already have assigned tasks. An exit works in the same way. A role, as an abstract description of a position in the system, is a fundamental concept in this class [23].

3 Organisation Design Problem

This research proposes to develop an *automated explicit organisational generator*. We hypothesise that it is possible to create roles from a *gdt* automatically. A *gdt* is a plan to achieve the main goal of the system, which includes operators that ensure that the decomposition satisfaction is equivalent to the main goal satisfaction [24].

In short, our proposal assigns goals to roles in a structured organisational chart taking into consideration some characteristics of the goals such as the ones that have the same parent goal, require the same skills to be performed, have a low predicted workload, etc. Additionally, design preferences can also determine whether to gather goals into a role or not, e.g., whether it is preferred a *flatter* or *taller* organisation; more *specialist* or *generalist* roles, if *matrix relations* are allowed or not, maximum *workload* per agent, etc. Moreover, the predicted throughput associated with a goal may indicate the need for the creation of new hierarchy levels and a *performer index* may imply that the same agent must, or must not, perform some goals.

For example, in a *gdt* for Printed Circuit Board (PCB) production, shown in Fig. 1a, the main goal is decomposed into two sub-goals: *Buy Supplies* and *PCB Assembly*. *Buy Supplies* also has two sub-goals: *Buy Components* and *Buy Other Supplies*. For these sub-goals, the skill *Purchase* is associated, which means that the agent(s) that will perform both *buy* sub-goals must be able to purchase items. The goal *PCB Assembly* has three sub-goals: *Apply Paste*, *Place Components*

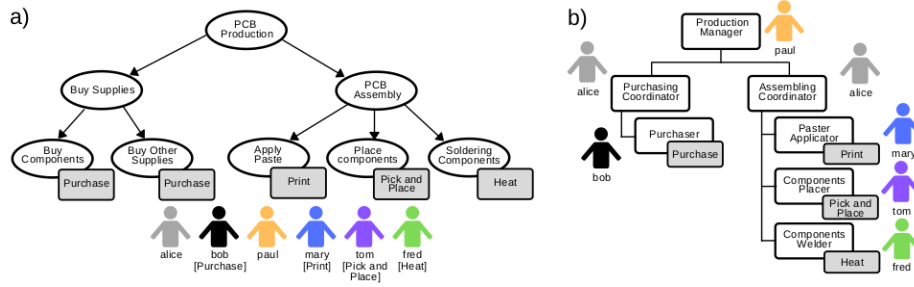


Fig. 1. Automated design for PCB Production. a) Inputs: goals tree and necessary skills. b) Output: organisational chart with the more generalist roles considering inputs.

and *Soldering Components*. The first is associated with the skill *Print*, the second with the skill *Pick and Place* and the latter with the skill *Heat*.

Fig. 1b shows a possible organisational chart based on the given *gdt* configured to be more *generalist*. In this example, the sub-goals *Buy Components* and *Buy Other Supplies* are assigned to the same role. In this sense, the same agent will perform both *Components* and *Other Supplies* purchases. This created role was placed below the *Purchasing Coordinator* role, as a subordinate.

However, one may ask: “is that solution the best one to choose?”. Still, there is no sufficient information to tell whether that structure is suitable or not. For instance, how many PCB’s are being produced per hour? How many different models are being produced? Are there other available resources? Any privacy requirement? These questions regard to varying situations in which the chosen structure depends on.

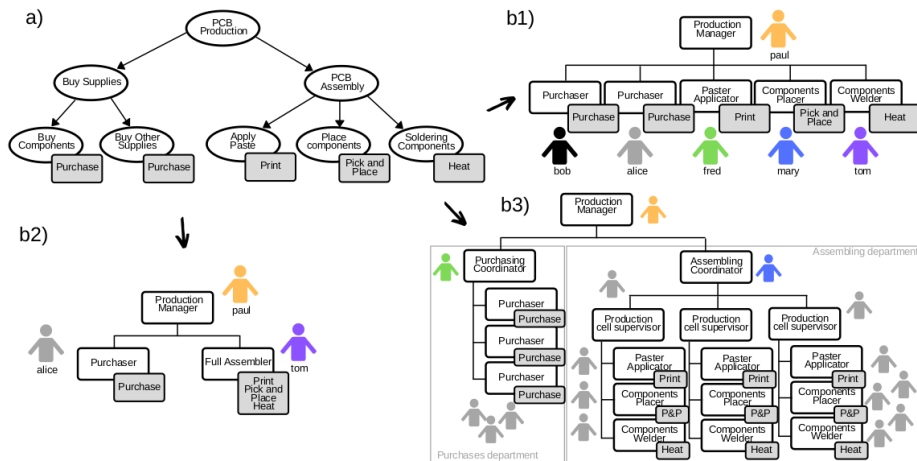


Fig. 2. Which organisational structure should be chosen?

Fig. 2 illustrates how diverse can the results be for the same given gdt . There is a solution in which only three roles were created in a very generalist and flat organisational structure. Another solution goes in the opposite direction, being very tall (hierarchical) and specialised. In fact, many aspects can influence organisational structures outcomes such as the chained sub-goals, agents' limited skills and goals fulfilment capacity, agents' communication capabilities and privacy needs, and so on [20]. Our proposal intends to address this problem by adding annotations to the goals to generate and choose a suitable organisational structure.

4 Proposed Method

We investigate the use of search algorithms to address the problem of creating and choosing an organisational structure. In this sense, the search space O is composed of all possible organisational charts $o \in O$. Each state o is composed of: (i) a set of role identifiers used in the organisational chart; (ii) the function gr for addressing the set of goals assigned to each role; (iii) the function pr for addressing the parent of each role which represent its immediate superior in the organisational chart where ϵ represents “no parent”, so that the root role r has $pr(r) = \epsilon$; and (iv) the function sr for mapping the set of skills in S which are associated with each role³.

$$\begin{aligned} o &= \langle R, gr, pr, sr \rangle \\ gr &: R \rightarrow 2^G \\ pr &: R \rightarrow R \cup \{\epsilon\} \\ sr &: R \rightarrow 2^S \end{aligned}$$

We can thus state that GoOrg searches for an organisational chart $o \in O$ that is suitable for a particular gdt . A gdt is composed of: (i) a set of goal identifiers G ; (ii) the function pg that returns the parent of each goal of the tree where ϵ represents “no parent”, so that the root goal g has $pg(g) = \epsilon$; and (iii) the function sg that addresses the set of necessary skills to achieve a given goal.

$$\begin{aligned} gdt &= \langle G, pg, sg \rangle \\ pg &: G \rightarrow G \cup \{\epsilon\} \\ sg &: G \rightarrow 2^S \end{aligned}$$

The difference of G and the set of goals assigned to roles is the set of not allocated goals nag , where:

$$nag(gdt, o) = gdt.G \setminus \bigcup_{r \in o.R} gr(r)$$

³ In a future work we will add other properties of goals and inputs for *GoOrg*.

4.1 State Transformations

All possible organisations populate the search space. To help the search for organisational charts, we define a transformation relation between two states. Fig. 3 represents each of the currently supported transformations in a respective area. Top and bottom of each area show respectively previous and final states. On each area, the graph on the left side is a *gdt* with three goals. Grey goals are the ones that were already assigned, and the black one is the goal that is being assigned. The graph on the right represents the roles of the organisational chart that is being created. The information between brackets describes the assigned goals, and eventually below it has the necessary skills to perform the respective role. Grey roles already exist, and the black one represents the role which is being explored for applying transformations.

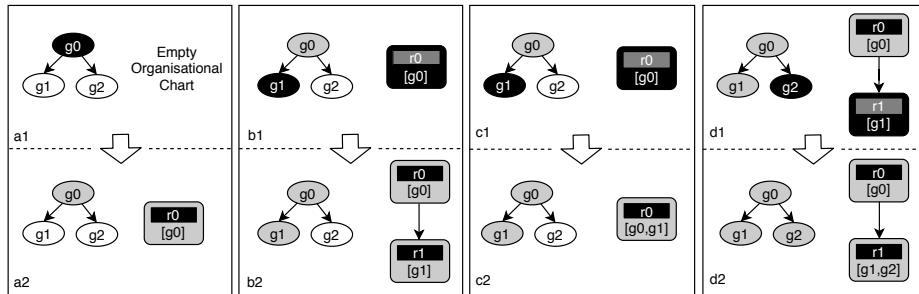


Fig. 3. Supported transformations.

In this illustration, we have: (a) the root goal generates the *root role*; (b) a sub-goal is assigned to a role to be subordinate of the role that contains its parent goal; (c) a sub-goal is vertically brought joining to the role that includes its parent goal; and (d) the sub-goal is horizontally carried joining to the role that contains its sibling sub-goal. For instance, the first transformation is illustrated on the area *a*, i.e., Fig. 3a shows the transformation of *a1* into *a2*. In this case, the root goal *g0* is going to be assigned and the organisational chart is empty as represented on part *a1*. After this transformation the chart has the role *r0* as represented by part *a2*. Considering the given input as $pg(g) = \epsilon$, the transformation for adding the root role is as follows:

$$o = \{\{\}, \{\}, \{\}, \{\}\}$$

$$addRootRole(g) \frac{}{o' = \{\{\mathbf{r}\}, \{\mathbf{r} \mapsto \{g\}\}, \{\mathbf{r} \mapsto \epsilon\}, \{\mathbf{r} \mapsto sg(g)\}\}}$$

On the area *b* of Fig. 3 the goal *g1* was assigned to a new role *r1* added as a subordinate of *r0*. This transformation is a possible process after the transformation displayed on area *a*. For this case, to add the role *r* as subordinate of *r'*

and allocate goal g to r , we require that g is a sub-goal of g' ($pg(g) = g'$) and g' is already allocated to r' ($g' \in gr(r')$) and have the following transformation:

$$o = \langle R, gr, pr, sr \rangle$$

$$\text{addSubordinate}(g, r') \frac{}{o' = \langle R \cup \{\mathbf{r}\}, gr \cup \{\mathbf{r} \mapsto \{g\}\}, pr \cup \{\mathbf{r} \mapsto \mathbf{r}'\}, sr \cup \{\mathbf{r} \mapsto sg(g)\}\rangle}$$

On area c the goal $g1$ was assigned to the existing role $\mathbf{r0}$ joining with the previously assigned goal $g0$. Again it can be illustrated from the state displayed on area a . In this case, there is no new role, the goal to be assigned is joined with a previously assigned goal $g0$. Formally, let the input be $pg(g) = g'$, and considering that $\{r' \mapsto g'\} \in gr$, the transformation for joining a subordinate is as follows:

$$o = \langle R, gr, pr, sr \rangle$$

$$\text{joinASubordinate}(g, r') \frac{}{o' = \langle R, gr \cup \{\mathbf{r}' \mapsto \{g\}\}, pr, sr \cup \{\mathbf{r}' \mapsto sg(g)\}\rangle}$$

Finally, on area d the goal $g2$ was assigned to the existing role $\mathbf{r1}$ joining with the previously assigned goal $g1$. This transformation can be applied from the state illustrated on area b . In this case, let the input be $pg(g) = g''$, there is a goal g' which parent is same, i.e., $pg(g') = g''$, and considering that $\{r'' \mapsto g''\} \in gr$ and $\{r' \mapsto g'\} \in gr$. In this sense, the transformation for joining a pair is as follows:

$$o = \langle R, gr, pr, sr \rangle$$

$$\text{joinAPair}(g, r') \frac{}{o' = \langle R, gr \cup \{\mathbf{r}' \mapsto \{g\}\}, pr, sr \cup \{\mathbf{r}' \mapsto sg(g)\}\rangle}$$

In fact, a goal can be assigned into a role in many ways. Currently, besides the parent relation of assigned goal(s), the associated necessary skills are also being taking into account. The parent is the way the algorithm use to assume relations among goals. A goal that is parent or a sibling of another potentially can be joined in the same role or it can be created as a close role, being a subordinate, according to the relation. The decision to join or not depends on the skills. The role skills must be compatible to be joined, which means, the role must already have the necessary skills of a goal candidate to be joined.

4.2 The Search Tree

To illustrate how the algorithm performs the search, Fig. 4 shows a gdt with three goals. There is a parent goal ($g0$) and two sub-goals ($g1$ and $g2$). To be fulfilled, $g1$ requires the skill $s1$. In the given gdt , two goals have no annotation. In case of $g1$, since it requires the skill $s1$, a role able to perform $s1$ can be

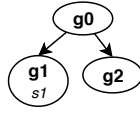


Fig. 4. Example of a simple goal decomposition tree (*gdt*).

assigned to other goals that also requires *s1* or does not require anything. Of course, a role that has no skills associated cannot perform the goal *g1*.

The algorithm creates and visits states, as illustrated in Fig. 5. The transformation of making the *root goal* be the *root role* of the organisational chart generates the first state. As expected, the first transformation has removed the element *g0* from the list of *to assign* goals, assigning it to the just created role called *r0*. The three possible successors of this state, is to add a role to assign *g1* as a subordinate of *r0*, add a role for *g2* as a subordinate of *r0* or even, bring up *g2* assigning it to *r0*, joining with other assigned goal(s) since their skills match.

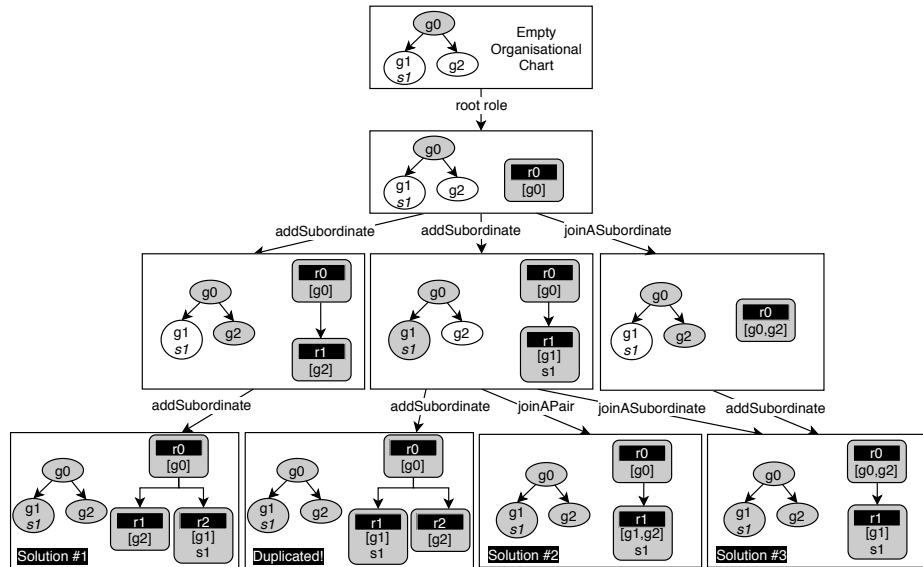


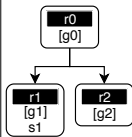


Fig. 5. Step by step of state search with all possible solutions for the given *gdt*.

Applying the transformations in the just created state on the left, where the goal *g2* was assigned to the role *r1*, it creates a role *r2* to assign *g1* putting it as a subordinate of *r0*. This is a target state since all goals were assigned successfully. This state is represented by the area with the label "Solution #1". The next area

on the right is a duplicated solution. Indeed, our method ignores the role name, using only assigned goals and parent relation to check redundancy, which is the case of solutions #1 and #2. Still, there are other solutions, as indicated by the other two areas.

Table 1 shows the referred solutions, or target states, generated by this method. The “Solution #1” is the most obvious chart, which is the generation of a role for each goal. The “Solution #2” is the result of joining horizontally the goals $g1$ and $g2$. It is possible because these goals are siblings and also the skills are compatible. The “Solution #3” is the result of joining vertically the goals $g0$ and $g2$. It was possible because the skills are compatible; in this case, both goals have no necessary skills. The arrows represent parent relations among goals.

Table 1. Organisational charts for a simple goals tree having a goal with an annotation

Solution	Chart	Description
#1		Organisational chart from adding two subordinates ($r1$ and $r2$) to the role $r0$. The same result would be achieved adding either $r1$ or $r2$ as subordinate of $r0$ and later add the other as a pair. This is the most specialised solution for the given goals tree.
#2		Organisational chart from adding $r1$ as subordinate of $r0$ and then joining the goals $g1$ and $g2$ into the role $r1$. It is possible because before assigning $g2$ the role $r1$ already had the skills needed by $g2$, which is actually nothing. The other way round would not be possible ($g2$ has not $s1$). It is one of the more generalist solutions for the given goals tree.
#3		Organisational chart from joining $g0$ and $g2$, since $g0$ has all the necessary skills needed by $g2$. Later $r1$ was added as a subordinate of $r0$. This solution is the more generalised and one of the more generalist solutions for the given goals tree.

In terms of hierarchy, i.e., the number of levels, all three solutions have the same height. In this case, it is not applicable any preference to choose a *flatter* or *taller* hierarchy. In terms of specialisation, “Solution #1” has more specialist roles, and the other solutions have more generalist roles for the given *gdt*.

Regarding the “Solution #2”, one may ask: why $g2$ joined with $g1$ and not the other way round? The reason is that a role created to perform $g2$ does not have any skills associated, and $g1$ needs the skill $s1$ to be performed. Since there is a sub-goal which has a skill associated, it was not possible to assign all the goals into a unique role. It would be the chart with more *generalist* roles and also the *flattest* solution since it would have assigned $g0$, $g1$ and $g2$ into an unique role.

4.3 The Search Algorithm

The proposed method for creating and choosing an organisational structure uses uninformed search also called blind search. We are using the well-known depth state-space search algorithm to illustrate how *GoOrg* is being implemented. As presented in Algorithm 1, it starts adding to a stack the given first state $o_0 \in O$.

Algorithm 1: Depth-limited Search

```

Data: Organisation  $o_0$ 
Result: Organisation
Stack  $n$ 
begin
   $n.push(o_0)$ 
  while  $n \neq \emptyset$  do
     $o \leftarrow n.pop()$ 
    if  $nag(gdt, o) = \emptyset$  then
      return  $o$ 
    end
     $n.push(successors(gdt, o))$ 
  end
  return null // failed on finding a goal state!
end

```

It represents the organisation that only has the *root role* created in the organisational chart R . The procedure, over and over, checks if the visiting state is a target state. When the tested state is not a target, the algorithm opens its successors to visit them later. The search ended when all the goals were assigned, i.e., $nag(gdt, o)$ is empty. The limit of this search, regarding the maximum depth of the tree, is G size, in this example it has three levels.

The function to get successors is illustrated in Algorithm 2. It is responsible for generating all possibilities for assigning a goal to roles. Indeed, as illustrated, the algorithm tries to place the goal to be assigned on each existing role applying the supported transformations. The $gr(r)$ function refers to the assigned goals for the specific role r , the same for the functions pr and sr .

The algorithms for transformations are roughly similar. The parent is eventually unknown because joining process may assign multiple goals into a unique role. For this reason, the algorithm tries to find the parent goal of the sub-goal to be allocated into the existing roles. Then $nag(gdt, o)$ is almost a copy, just skipping the current goal. Later the R is copied and also is updated with the just created role. Finally, this new or modified role is considered a possible successor state for further searches.

In the previous example, as illustrated in Fig. 5, all the possible solutions are being shown. However, the algorithm stops after finding the first solution, which remarks on the importance of ordering. The solutions are sorted by *cost*

Algorithm 2: successors

```

Data: List  $\langle G, pg, sg \rangle$   $gdt$ , Organisation  $o$ 
Result: List  $\langle$  Organisation  $\rangle$ 
begin
  List  $suc$ 
  foreach Goal  $g$  of  $nag(gdt, o)$  do
    foreach Role  $r$  of  $o.R$  do
      if  $gr(r)$  contains  $pg(g)$  then
         $addSubordinate(r, suc, g)$  // Add as a child role
        if  $sg(g) \in sr(r)$  then
           $joinASubordinate(r, suc, g)$  // Join  $g$  into *this* role
        end
      else if  $pg(g) \in gr(pr(r))$  and  $sg(g) \in sr(r)$  then
         $joinAPair(r, suc, g)$  // Join goal  $g$  into *this* role
      end
    end
  end
  return  $suc$ 
end

```

functions which are related to the user preferences. For instance, if a more *generalist* structure is preferred so “pair roles” creation is costly, and joining pairs is cheaper. It makes preferable a chart with fewer pairs as possible.

5 Future work

For the next step of our research, the designing process is being split into two phases: the *organisation design* and the *resource allocation* process. With this separation, it is expected that *GoOrg* becomes more suitable to deal with asynchronous changes on the system’s resources availability and redesign requests.

On the next step, still on designing process, we will add new inputs such as *predicted workload*, *necessary resources*, *performer index*, *communication topics*, and *predicted throughput*. The *predicted workload* can be used to know how many agents should take the same role or if the same agent can perform more than one role. The *performer index* indicates that the same agent must perform some goals and, contrarily, can tell that two goals cannot be performed by the same agent, for instance in a process in which something is made and must be verified by another agent. With *communication topics* and *predicted throughput*, the hierarchy levels and departmentalisation can be set. These data may also allow enhancing the algorithm to decide when a coordination role can be subtracted, maintained or even new ones created. Other state-space search algorithm and cost functions will be experienced for optimisation purpose and to give more possibilities in terms of structures.

In the sequence, we plan to develop the second process, i.e., *resources allocation*. This process will bind resources and roles. The inputs are *available*

agents and skills, available artefacts and organisation design preferences. This allocation process aims to guarantee that the created structure is viable, i.e., can be well-formed when it runs with the given resources. Finally, the output is an organisational chart with artefacts allocated and agents assigned to roles.

The allocation process can solve some challenges that do not require a re-design. To illustrate it, back to *PCB Production* example, consider that *Buy Components* sub-goal also needs *Electronics Knowledge* skill and the chart has created different roles for purchasing, they can be called *Components Purchaser* and *Other Inputs Purchaser*. Consider that *agent A* and *agent B* play, respectively, the referred roles having all the necessary skills to play both. Consider now that *agent A* left the system and *agent C* has joined it, but this agent has no *Electronics Knowledge* skill. The resource allocation process can move *agent B* to *Components Purchaser* role, assigning *agent C* to *Other Inputs Purchaser* role.

It is also expected to make *GoOrg* suitable to deal with asynchronous changes on the system's resources availability and redesign requests. For instance, with simple changes in the availability of resources, the process can be lighter. However, with more significant changes, for example, on the *gdt*, a complete redesign process may be necessary, a function that can be triggered by the allocation phase. In this solution, as illustrated in Fig. 6b1, the goals were centralised in a unique role which is more *generalist* to achieve more goals with sometimes different associated skills.

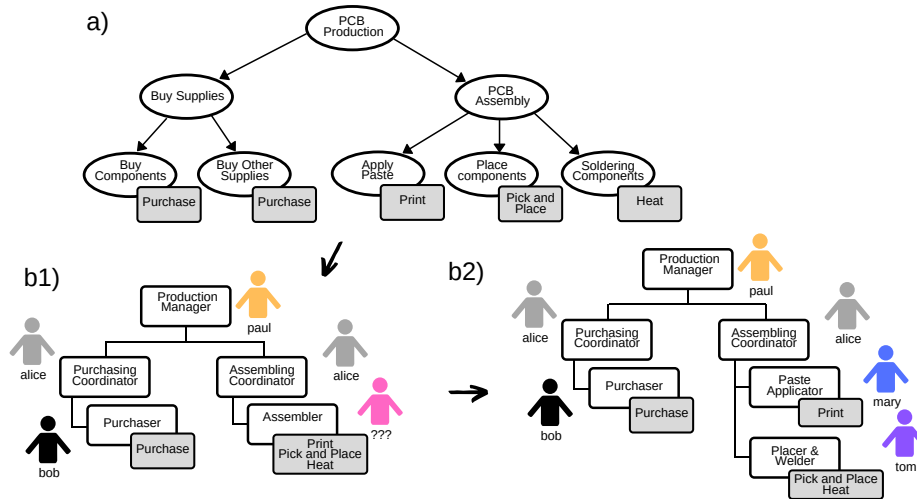


Fig. 6. More generalist organisation chart for the given goals and available agents.

However, the illustration also exemplifies a situation where there is no available agent with all necessary skills to perform the role *Assembler* since it is

gathering the skills *Print*, *Pick and Place* and *Heat*. Fig. 6b2 shows a possible solution assuming that an agent *mary* is able to perform *Print* and an agent *tom* can perform *Pick and Place* and *Heat*. In this case, the more *generalist* well-formed organisation is represented by this last chart.

We also expect to create other organisational aspects as outputs, i.e., *Organisational Scheme* and *Organisational Norms*. The former refers to sets of goals allocated to different roles that should be performed by the same agent in a specific sequence. The latter regards especially to general obligations, such as, when adopting a role created by the method *GoOrg* the agent is obligated to perform the missions associated with the referred role.

Finally, we will evaluate our solution using existing domains [3, 17, 26]. We will first assess the number of input parameters needed by *GoOrg*. With these inputs, we will evaluate the ability of *GoOrg* to design organisations properly. We will vary aspects of the simulated domains presenting them as more static or dynamic, with shorter or longer goals, with more chained or independent goals, etc. These domains will be used to experiment with different user preference parameters.

From a literature perspective, we can select an organisational structure by its features as the potentially better solution for the given problem and scenario. Besides testing this candidate, other organisational structures will be created for comparison purposes. It is expected to fulfil all the goals in less time with the best candidate. The results of the simulations should give us insights to discuss literature perspectives, the adhesion of our method and simulation with literature and potentially *GoOrg* application as a *testbed* for organisational structures.

Among the assumptions we want to evaluate, we have: how the span of control affects the effectiveness of the organisation [13] varying the height of the hierarchy to check the impact on agents communication and coordination [14, 30]. In this sense, we can check whether a few number of levels really can lead to faster decisions and lower overhead costs [14, 30] and if highly structured organisations are best for repetitive operations [13].

6 Related Work

In the administration area, there are many studies about organisation design, including some frameworks that may help companies and other organisations to design their structures [2, 7]. In multi-agent systems, we usually have *manual organisational generators*, i.e., approaches that allow a human to design organisations in a wide variety of structures and other aspects as norms, roles, relations, organisational goals and ontologies, e.g., Moise+ [18], THOMAS [5], STEAM [31] and AALADIN [12].

In spite of having many studies about organisation design, there are still many gaps regarding the full range of disciplines and high complexity of organisations. Considering only automatic organisation generators, the focus of this research, there are few studies.

Automated planning is a research area that has produced many contributions to MAS design. When developing planners for multiple agents, the organisation design is an intrinsic outcome. Some examples of planners able to generate organisations are TÆMS [8] which provides a way to quantitatively describe individual tasks which are performed in shared environments, DOMAP [3] which is a decentralised MAS task planning and Sleight’s agent-driven planner [27] using a decentralised Markov Decision Process Model.

Considering bottom-up approaches So [29] did one of the earlier researches on Multi-Agent Systems organisation design. This study over the characterisation of different organisation designs, including self-organised ones and the reconfiguration process for stable organisations. There are several studies over self-organised swarms which use very computationally limited agents [19], and there is no complex coordination mechanism among agents [32].

In the class we have positioned our research, we found only a few works: SADDE [23] and ODML [17], which are algorithms that use as input mathematical models to predict efforts and create an organisational structure; MaSE-e [10] which is a method for creating organisation structures extending the engineering method MaSE; and KB-ORG [25, 26] that takes goals and roles to bind agents and create coordination levels. Although seminal, we think the methods have challenges to overcome, especially regarding inputs in which we are proposing a method to produce *roles* in a way to make inputs easier to handle.

Table 2 gives an overview of *explicit organisation generators* we have found⁴. We are comparing a few features related to inputs, intrinsic features and outputs. The first columns refer to inputs. We start checking whether *goals are inputs* since it gives an idea of the start point of each approach. The *no need roles as inputs* indicates if the generator needs this input. The column *Bound Agents are inputs* represents the capability of the generator to receive as inputs a structure earlier created with bound resources.

The next columns represent features of the generators. The column *has quantitative analysis* describes the capability of the generator to assess the goals creating structures that take into account quantitative parameters such as goal expected needed effort to be performed. *Organisations are explicit* refers to methods that use explicit organisation representations. *Is domain-independent* relates to methods that are suitable for any problem domain.

The next columns are related to the main outputs of the generators. *Creates roles* refers to the ability to automatically create roles, combined with *roles are inputs* says whether the approach uses or not the concept of roles. The *creates coordination levels* column represents the ability of the method to create coordination roles according to coordination needs automatically. *Create viable organisations* represents the ability of the generator to check available resources to create organisations that can be fulfilled when running. *synthesise organisational norms* inform whether generators are automatically creating organisa-

⁴ Legend: (Y)es, (-)No, On (R)oadmap and (*) comments. Table comments: *1 The output is a nodes tree, not exactly an organisational chart. *2 There is no hierarchy.

Organisation Generator	Goals are inputs	No need roles as inputs	Bound Ag. are inputs	Has quantitative analysis	Organisations are explicit	Is domain-independent	Creates Roles	Creates Coord. Levels	Creates viable org.	Synthesise Org. Norms	Bind agents and roles	Creates departments	Represents roles in a chart	Does state reorganisation	Does structure reorg.
GoOrg	Y	Y	Y	Y	Y	Y	Y	Y	Y	R	Y	R	Y	R	R
SADDE	Y	-	-	Y	Y	Y	-	-	Y	-	Y	-	-	-	-
MaSE-e	Y	-	Y	Y	Y	Y	-	-	Y	-	Y	-	*2	Y	R
KB-ORG	Y	-	-	Y	Y	Y	-	Y	Y	-	Y	-	Y	-	-
ODML	Y	-	-	Y	Y	Y	-	-	Y	-	Y	-	*1	-	-

Table 2. Comparison among organisation generation methods.

tional norms. *Bind agents and roles* tells whether the method is doing agents allocation job or not.

The next columns regard to byproducts of the generators. *Creates departments* refers to the specific ability of the generator to create organisational departments automatically. *Represents roles in a chart* relates to methods that represent organisations as usual organisational charts.

The following columns are related to the capability of the generators to deal with reorganisations. *Does state reorganisation* refers to the ability to move agents from some responsibility to another without needing to trigger a restructuring process. *Does structure reorganisation* refers to the ability to create new structures based on an old one.

Finally, as we agree with many authors that there is no single type of organisation suitable for all situations [16], we also recognise that there is no individual approach ideal for creating all organisations [6]. In both cases, each offers some advantages that the others may lack, especially regarding different organisation generator classes. In the presented comparison, we tried to show an overview of those organisation generators based on the assumption that explicit organisational structures can provide advantages on designing open systems.

7 Conclusion

This paper has presented a proposal for an automated generator of explicit organisations based on goals and annotations as inputs. The current status of this research shows that it is feasible to draw an organisational chart using as input organisational goals with some annotations such as necessary skills to perform each goal. It is intended to enhance the current version of our method adding new inputs to bring necessary information to produce useful organisa-

tional charts, taking advantage of opportunities to join goals on the same roles, adding or removing coordination levels. According to performance issues, we can add heuristics to improve the search algorithm.

We have also presented our classification regarding related research of automated organisation generators: (i) *automated organisation design by task planning*; (ii) *self-organisation* approaches; and (iii) *automated explicit organisation generators*. It shows that different strategies address the challenge of *organisation design*. The approaches have advantages and drawbacks being more suitable according to the system's purpose and environment conditions. Besides, we think that each class gives some contribution and a combination of them can lead to a comprehensive MAS design.

Besides the organisational chart creation itself, an extra outcome of *GoOrg* is a proposition of a model that identifies different designing phases done by various methods which potentially can be used together to design a whole MAS. Indeed, when splitting *GoOrg* to fit this model, we could identify that our method is actuating on two processes: organisation design and resources allocation. The allocation of resources done before the execution is a guarantee that when running the created organisational chart can be filled by the available resources, i.e., can be a well-formed organisation.

About evaluation criteria, it is intended to apply the model in known domains testing if it can build suitable structures. These organisations will be simulated in a variety of conditions and checked if goals were fulfilled. By tuning preferences, it is expected to create better arrangements for our testing domains. We also intend to compare the results in terms of time to accomplish the goals between the best candidate and other organisational structures.

References

1. Amaral, C.J., Hübner, J.F.: Goorg: Automated organisational chart design for open multi-agent systems. In: De La Prieta, F., González-Briones, A., Pawleski, P., Calvaresi, D., Del Val, E., Lopes, F., Julian, V., Osaba, E., Sánchez-Iborra, R. (eds.) PAAMS. pp. 318–321. Springer International Publishing, Cham (2019)
2. Burton, R.M., Obel, B., Desanctis, G.: Organizational design: a step-by-step approach. Cambridge University Press (2011)
3. Cardoso, R.C., Bordini, R.H.: A modular framework for decentralised multi-agent planning. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 1487–1489. São Paulo, Brazil (2017)
4. Cardoso, R.C., Bordini, R.H.: Decentralised Planning for Multi-Agent Programming Platforms (Aamas), 799–807 (2019)
5. Criado, N., Argente, E., Botti, V.: THOMAS: An agent platform for supporting normative multi-agent systems. *Journal of Logic and Computation* **23**(2), 309–333 (2013)
6. Daft, R.L.: Organization Theory and Design. South-Western College Pub, Centage Learning, 10th edn. (2009)
7. De Pinho Rebouças De Oliveira, D.: Estrutura Organizacional: Uma Abordagem Para Resultados e Competitividade. ATLAS EDITORA (2006)

8. Decker, K.S.: Environment Centered Analysis and Design of Coordination Mechanisms. PhD Thesis, University of Massachusets (May 1995)
9. DeLoach, S.A.: Modeling organizational rules in the multi-agent systems engineering methodology. In: Cohen, R., Spencer, B. (eds.) *Advances in Artificial Intelligence*. pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
10. DeLoach, S.A., Matson, E.: An Organizational Model for Designing Adaptive Multi-agent Systems. The AAAI-04 Workshop on Agent Organizations: Theory and Practice (AOTP 2004). pp. 66–73 (2004)
11. DeLoach, S.A., Oyenon, W.H., Matson, E.T.: A capabilities-based model for adaptive organizations. In: *Autonomous Agents and Multi-Agent Systems*. vol. 16, pp. 13–56 (2008)
12. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. *Proceedings - International Conference on Multi Agent Systems, ICMAS 1998* pp. 128–135 (1998)
13. Fink, S., Jenks, R., Willits, R.: *Designing and Managing Organizations*. Irwin Series in Financial Planning and Insurance, R.D. Irwin (1983)
14. Galbraith, J.R.: *Designing organizations: an executive briefing on strategy, structure, and process*. Jossey-Bass Publishers - San Francisco (1995)
15. Hatch, M.: *Organization Theory: Modern, Symbolic, and Postmodern Perspectives*. Oxford University Press (1997)
16. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. *Knowledge Engineering Review* **19**(4), 281–316 (2004)
17. Horling, B., Lesser, V.: Using quantitative models to search for appropriate organizational designs. *Autonomous Agents and Multi-Agent Systems* **16**(2), 95–149 (2008)
18. Hübner, J.F., Sichman, J.S.: Organização de sistemas multiagentes. III Jornada de MiniCursos de Inteligência Artificial JAIA03 **8**, 247–296 (2003)
19. Labella, T.H., Dorigo, M., Deneubourg, J.L.: Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems* **1**(1), 4–25 (2007)
20. Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., Colombo, A.W.: Smart agents in industrial cyber physical systems. *Proceedings of the IEEE* (2016)
21. Mintzberg, H.: The design school: Reconsidering the basic premisses of strategic management. *Strategic Management Journal* **11**(May 1989), 171–195 (1990)
22. Pattison, H.E., Corkill, D.D., Lesser, V.R.: Chapter 3 - instantiating descriptions of organizational structures. In: Huhns, M.N. (ed.) *Distributed Artificial Intelligence*, pp. 59 – 96 (1987)
23. Sierra, C., Sabater, J., Augusti, J., Garcia, P.: SADDE: Social agents design driven by equations. *Methodologies and software engineering for agent systems*. Kluwer Academic Publishers pp. 1–24 (2004)
24. Simon, G., Mermet, B., Fournier, D.: Goal decomposition tree: An agent model to generate a validated agent behaviour. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) *Declarative Agent Languages and Technologies III*. pp. 124–140. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
25. Sims, M., Corkill, D., Lesser, V.: Knowledgeable Automated Organization Design for Multi-Agent Systems. *Challenge* pp. 1–42 (2007)
26. Sims, M., Corkill, D., Lesser, V.: Automated organization design for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* **16**(2) (2008)
27. Sleight, J., Durfee, E.H.: Organizational design principles and techniques for decision-theoretic agents. In: *Proceedings of the 2013 International Conference on*

- Autonomous Agents and Multi-agent Systems. pp. 463–470. AAMAS '13, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2013)
28. Sleight, J.L., Durfee, E.H., Baveja, S.S., Cohn, A.A.E.M., Lesser, E.V.R.: Agent-Driven Representations, Algorithms, and Metrics for Automated Organizational Design (2015)
 29. So, Y.P., Durfee, E.H.: Chapter X. Designing Organizations for Computational Agents (1996)
 30. Stoner, J., Freeman, R.: Management. Prentice-Hall (1992)
 31. Tambe, M.: Towards Flexible Teamwork. *Journal of Artificial Intelligence Research* **7**, 83–124 (1997)
 32. Ye, D., Zhang, M., Vasilakos, A.V.: A Survey of Self-organisation Mechanisms in Multi-Agent Systems. *IEEE Transactions On SMC: Systems* **47**(3) (2016)