Applying Multiagent Systems for scheduling Multiprocessor Task Problems

Matthijs Brouns

Delft University of Technology
Faculty of Technology, Policy, and Management
Delft, The Netherlands,
m.brouns@student.tudelft.nl

Abstract Multiagent Systems are a promising method for planning and scheduling when dealing with multi stakeholder preferences. Most literature considering MAS for scheduling has focused on their application for traditional Job-Shop problems, without covering other scheduling problems such as Multiprocessor Task problems, scheduling problems in which jobs require multiple processors simultaneously. This article describes a literature study in which three MAS architectures (Problem Decomposition, House Allocation and Stable Matching, and Combinatorial Auctions) have been assessed on their applicability for MPT problems. Problem Decomposition is able to model the MPT problem, but is not able to incorporate agent preferences. We propose an algorithm which is obtained by combining two extensions to the stable Matching Problem and the House Allocation Problem, namely the HAP with existing tenants, and the Student/Project Allocation Problem. Combinatorial auctions are well-equipped to model the MPT problem with minor extensions to the bidding language. However, their design space to ensure truthfulness and strategy-proofness is greatly diminished in domains where payments between agents are not possible.

Keywords: Multiagent Systems, Scheduling, Job-shop, Multiprocessor Tasks

1 Introduction

Ever since their introduction in the 1970s, the use of Distributed Artificial Intelligence (DAI) and Multiagent Systems (MAS) has grown in both the number as well as the diversity of applications. One of the domains in which MAS have been numerously and successfully applied is the domain of planning and scheduling. Applying MAS for planning and scheduling can have many benefits in for example flexibility in multi-objective optimization (Agnetis, Billaut, Gawiejnowicz, Pacciarelli, & Soukhal, 2014), composability, reusability and distribution (Jennings & Bussmann, 2003).

Scheduling is a decision-making process in which tasks are allocated to resources over a time period with as a goal to optimize one or more objectives

(Pinedo, 2012). The domain of scheduling consists of a virtually unlimited number of problem types (Brucker, 2007, p. 1), of which the most common is the traditional job-shop problem. A job-shop problem is characterized by a set of jobs (or tasks) $J = \{j_1, j_2, ..., j_n\}$ which need to be scheduled on m identical machines (or processors), while minimizing make-span. A considerable amount of literature has been published on the application of MAS for job-shop scheduling problems (Gabel & Riedmiller, 2008; Qing-song & Li-ming, 2009; Siekmann, Hartmanis, & Leeuwen, 2002; Wu & Weng, 2005; N. Liu, Abdelrahman, & Ramaswamy, 2005; Kouider & Bouzouia, 2012; Lee & Kim, 2008). However, when reviewing the literature it was found that previous studies in this domain have so far neglected many other types of scheduling problems, one of which being scheduling problems with multiprocessor tasks (MPT). In this type of problem, a set of job needs to be scheduled on machines, but requires multiple machines simultaneously for processing.

The goal of this paper is to assess the applicability of MAS scheduling for MPT scheduling problems. To achieve this goal, this paper is split into three parts. This paper begins by describing the applied research methodology. It will then go on with describing JMPT problem definitions in detail and proposes a comparison framework for assessing the applicability of MAS for this problem. The subsequent chapters go into detail on several solution strategies and will assess these based on the earlier described comparison framework. The paper ends with a concluding section which will discuss the main findings, as well as propose directions for future research.

2 Research methodology

The goal of this research is to assess the applicability of MAS for application in the scheduling of Multiprocessor Task problems and to critically reflect on their strengths and weaknesses in a variety of cases. In order to obtain the required information a systematic literature review was conducted. First, an exploratory analysis of MAS scheduling and MPT scheduling was performed by searching the TU Delft Library, Scopus, and Web of Science using the keywords *MPT*, *Multi-Processor Tasks*, *Multiagent*, *MAS*, and *Agent-based*. Papers were selected by filtering on journals with an operations research or computer science background such as *Artificial Intelligence*, *European Journal of Operational Research*, and *Journal of Scheduling* as well as conference proceedings from conferences such as *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. This was done mainly because apart from MAS there are several other research domains which focus on applying agents, such as agent-based systems and agent-based simulation. References from articles were used to search for the theoretical foundations underlying the systems which were found in (Wooldridge, 2002; Russel & Norvig, 2009; Cramton, Shoham, & Steinberg, 2006). From this exploratory analysis it became clear that there are a number of main Multiagent System Architectures used for the allocation of resources among self-interested agents. This research focuses on the House Allocation Problem (HAP), the Sta-

ble Matching Problem (SMP), and Combinatorial Auctions (CA) as these are most commonly applied MAS architectures.

In order to assess the applicability of Multiagent Systems for MPT problems, a literature study was performed to find the characteristics of these problems. These characteristics were found in (Brucker & Krmer, 1995; Brucker, 2007; Pinedo, 2012). Based on these characteristics, a comparison framework will be proposed using which the applicability of different Multiagent System architectures can be identified.

3 Multiprocessor Task Problems

Scheduling problems can be formally defined using a $\alpha|\beta|\gamma$ notation as introduced by (Graham, Lawler, Lenstra, & Kan, 1979), where α denotes the machine characteristics of the job, β the job characteristics, and γ the optimality criterion.

In contrast with the most common types of scheduling problems, in which a job is processed by at most one machine at a time, systems with multiprocessor tasks require jobs to be processed by multiple machines simultaneously. This means, that given m different machines $M = \{m_1, ..., m_m\}$ and j jobs $J = \{j_1, j_2, ..., j_j\}$, each task $i \in J$ requires all processors belonging to a subset $\mu_i \subseteq \{m_1, ...M_m\}$ for a processing period p_i. Jobs requiring the same machine cannot be processed simultaneously, and are called incompatible. If jobs do not share processors, they are called compatible (Jansen & Porkolab, 2000).

There are four main types of MPT problems (Brucker, 2007, p. 317). In a *general shop problem with Multiprocessor Tasks* (GMPT), each job i has n_i multiprocessor tasks $\{O_{i1}, ...O_{i,n_i}\}$ with processing times p_{ij} and processors $\mu_{ij} \subseteq \{m_1, ..., m_m\}$. Task O_{ij} requires each processor μ_{ij} for time p_{ij} to be completed.

The second type, a *job-shop problem with multiprocessor tasks* (JMPT) is a special case of this (GMPT) problem in which there is a distinct order $\succ$ in its set of tasks $\{O_{i1}, ...O_{i,n_i}\}$

Thirdly, The *flow-shop problem with multiprocessor tasks* (FMPT) is once again a special case of the (JMPT) problem in which there is a strict order of operation to be performed on all jobs, all jobs are processed in the same order by the same sets of machines ($\mu_{ij} = \mu_j \subseteq \{m_1, ..., m_m\}$ for all tasks $j = 1, ..., r$), and no pair of tasks belonging to a single job can be processed simultaneously.

Finally, *open shop problems with multiprocessor tasks* (OMPT) are defined as FMPT without precedence constraints between tasks.

In scheduling problems with multiprocessor tasks, the α is set to JMPT, GMPT, FMPT, or OMPT and an extra m symbol is added in case the number of processors for each task is fixed.

Typically, all MPT problems which do not either have two machines, or completely identical machines are part of the NP-hard complexity category, which means they cannot be solved optimally in polynomially bounded time (Brucker, 2007, p.343).

The job characteristics of a scheduling problem are defined by a set containing at most five elements:

- β_1: Set to *pmtn* (pre-emption) when jobs can be interrupted and resumed;
- β_2: Describes precedence relations between jobs;
- β_3: Specifies release dates for the jobs;
- β_4: Specifies additional job-dependent set-up times by setting $\beta_4 = s_{ij}$;
- β_5: Specifies hard deadlines.

Typical optimality criteria in scheduling problems are minimising makespan, minimising total flow time, and minimising tardiness. These objectives are global objectives, in the sense that they attempt to give a desirability of the overall allocation. When considering the application of Multiagent Systems for scheduling problems, we are typically less interested in overall schedule optimization functions, but more in the desirability of the schedule for the individual agent. For this article, we assume the jobs to have a strictly ordered preference over the possible allocations, and the goal of the scheduling problem is to find a Pareto optimal solution given this ordering. Apart from the cases where the jobs have a preference ordering over the machines, it is also possible for the machines to have a preference ordering over the jobs. These types of problems will also be considered in this article.

4 Comparison Framework

Now that the scheduling problem discussed in this article is formally defined, this article will now move on to discuss the comparison framework used for assessing the applicability of the Multiagent System architectures. There are three main aspects on which HAP, SMP, and CA will be assessed, each consisting of several elements. These aspects are the capability of the architecture to represent the problem, the possibility for incorporating the various job characteristics in the architecture, and the flexibility for incorporating agent preferences.

The capability of the architecture to represent the scheduling problem consists of two elements. All discussed architectures are instances of agent allocation architectures, rather than scheduling architectures. Allocation problems differ from scheduling problems in that instead of allocating resources over a time period, they simply allocate the resources once. Assessing whether the allocation methods can be altered to support scheduling is the first element of this aspect. The second element is assessing whether the architecture is able to represent the inherent feature of multiprocessor tasks that the tasks require multiple subsets of machines, which can partially overlap. The third and final element deals with whether the allocation methods can be used to represent the specific GMPT, JMPT, FMPT, and OMPT problems described earlier.

The second main aspect is the capability of the architecture in incorporating the discussed job characteristics, such as the pre-emption of jobs, setting deadlines, and specifying release dates.

The final main aspect the architectures will be assessed on is the flexibility they offer when implementing agent preferences. Elements in this aspect include whether its possible for the machines to elicit preferences, and whether macro-level allocation preferences can be elicited.

A visual summary of this comparison framework is displayed in Figure 1.

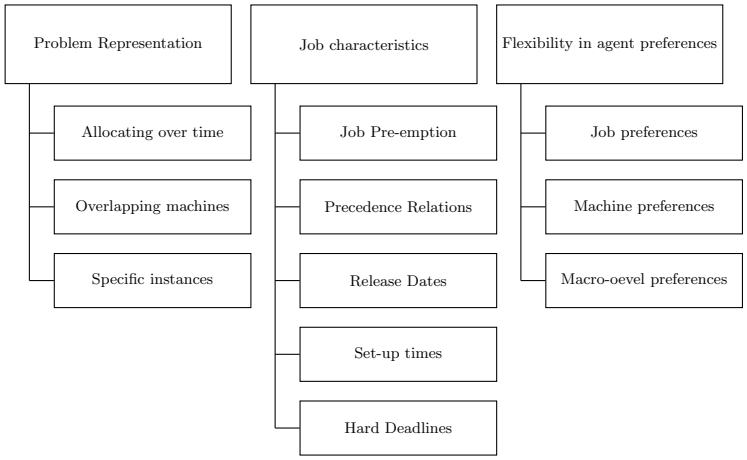


Figure 1: Comparison Framework

5 Problem Decomposition

Problem Decomposition is a cooperative scheduling solution approach in which the problem space is divided into independent subsets which can be solved using only local information. Problem decomposition is applied by a number of authors for idle time minimization in regular job-shop problems (Wang & Liu, 2006; Kouider & Bouzouia, 2012).

The proposed problem decomposition approaches are based on representing the problem as a disjunctive graph and finding cliques in this graph. These cliques are then represented by agents which can solve their local scheduling problem relatively independent of the other cliques. The local scheduling problem can be solved using a number of methods such as greedy heuristics or more advanced optimization algorithms.

Kramer (1995) proposes in his work several modifications on the disjunctive graph model for it to be capable of representing multiprocessor task problems, making the same problem decomposition techniques feasible for MPT problems. However, in practice it is expected that the found cliques in MPT problems are much larger as there are typically more dependencies between machines in these types of problems.

Since a disjunctive graph is capable of representing all job characteristics described in the comparison framework, it is possible for problem decomposition to incorporate these without problems.

With regards to the flexibility in agent preferences we quickly see that, while this solution method is proposed in the literature as a MAS architecture, it refers to different types of agent than are meant in this article. The agents used in problem decomposition are computational helpers, used to solve a problem in a divide-and-conquer style. The agents described in this article however, refer to agents which can be considered as clients to the scheduling solution and have clear preferences over the allocations. Therefore, problem decomposition fails to provide flexibility with regards to agent preferences.

In conclusion, it becomes clear that while problem decomposition can be a suitable strategy for solving regular multiprocessor task problems, it is not suitable in a case where agents have clear preferences over the allocations. Furthermore, its applicability is further hindered since there are typically more dependencies between processors in the case of multiprocessor task scheduling.

6 House Allocation & Stable Matching

The House Allocation Problem is an instance of a single-sided allocation problem. The HAP is characterized by a tuple $(A, H, \succ)$ in which A is a set of agents, H is a set of resources (or houses), and each agent $a \in A$ has a strictly ordered preference over the resources. A solution to the HAP is an allocation in which each agent receives exactly one resource, and no two agents can be made better off when switching their allocated houses.

The Stable Matching problem is a very similar type of problem but is instead a two-sided allocation problem, in which not only the agents A have preferences over the resources H, but the resources also have preferences over the agents. A solution to the SMP is an allocation in which all each agent receives one house, each house receives one agent, and no house and agent can be made better off when trading.

The default algorithm to solve the HAP is applying Serial Dictatorship, which specifies an order over the agent, and in this order let each agent receive his highest preferred house among the still available resources. This algorithm is easily proven to be strategy-proof and Pareto optimal (Ergin, 2000, p.6).

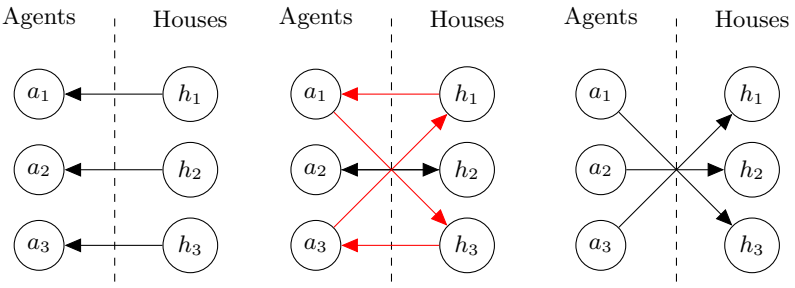
The typical algorithm for solving the SMP is the Gale-Shapley algorithm which guarantees that all agents get matched and all matches are stable. This algorithm works in several steps (Gale & Shapley, 1962):

1. All agents A propose to their preferred resource H;
2. The agents in set H reply "maybe" to their preferred proposal, and "no" to all others;
3. Due to the provisional nature of the "maybe" response, the agents in set H are allowed to "trade up" in case they get a better offer in a next iteration;
4. Repeat until all agents are matched.

The House Allocation problem typically refers to a single-time matching problems in which the agents and houses do not vary over time, making it unsuitable for scheduling tasks. To deal with the allocation of houses over time, an extension to this problem is available called the house allocation problem with existing tenants (Abdulkadiroglu & Sonmez, 1999). This extension is based on the problem of allocating dorm rooms to college students and is capable of solving the house allocation problem in a situation in which freshman arrive and want new houses, and graduates leave and vacate houses. Abdulkadiroglu and Sonmez (1999) introduce in their work an algorithm inspired by Gale's top trading cycles algorithm (Figure 2) (Gale & Shapley, 962) which works as following:

Algorithm 1.1: Top Trading Cycles with existing tenants

```
1  Let G be a bipartite graph with a set houses H and agents A
2  Let h(a) be the owner of house h
3  Let ≺(a) be the preference order over the houses H of agent a
4  Let π be a priority order over the agents
5
6  While |A| >= 1 && |H| > 1 do:
7    if h(a) = \null do:
8      draw edge in G from h ∈ H to π(0)
9    else do:
10     draw edge in G from h ∈ H to h(a)
11   end if
12   draw edge in G from a to ≺(a)
13
14   remove all cycles by assigning ≺(a) to a
15   Remove assigned h from H
```



(a) Houses point to owners

(b) Agents point to houses, a cycle forms

(c) Trade the houses in a cycle

Figure 2: Top Trading Cycles

This algorithm reduces to a serial dictatorship algorithm in the case where there are no existing tenants, and to the top trading cycles in case there are no

new tenants. A similar algorithm using trading cycles is described in (Jalilzadeh, Planken, & De Weerdt, 2010) for scheduling barges onto locks.

An alternative algorithm to find the outcome of the top trading cycles mechanism is a *You request my house - I get your turn* (YRMH-IGYT) algorithm. In this algorithm, for any given ordering f, the first agent f_1 is assigned his most preferred house, then the second agent f_2 is assigned his highest preference house among the remaining houses. This continues until an agent f_n requests the house of an existing tenant f_m. If, in that case the existing tenant is already assigned a new house, the agent f_n is assigned the requested house. Else, the remaining ordering of the agents is altered to have the existing tenant f_m be at the top of the order. If a cycle forms where each agents demands the house of the next agent, all houses are traded. This algorithm is proven to be the same as the top trading cycles algorithm (Abdulkadiroglu & Sonmez, 1999, p. 251).

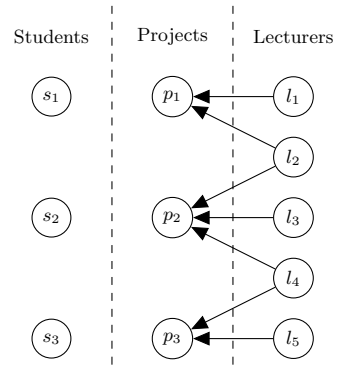


Figure 3: Student/Project allocation problem

In order to make these allocation problems suitable for Multiprocessor task scheduling, it should be possible for the agents to acquire multiple houses. This situation is described earlier in the literature as the Student / Project Allocation problem (SPA) (Abraham, Irving, & Manlove, 2007; Iwama, Miyazaki, & Yanagisawa, 2012; Kwanashie, Irving, Manlove, & Sng, 2014), which both exists in a single-sided preference as well as a two-sided preference implementation (Figure 3). An instance of the SPA can be defined as follows. Let $S = \{s_1, ..., s_n\}$ be a set of students, $P = \{p_1, ..., p_m\}$ a set of projects, and $L = \{l_1, ..., l_q\}$ a set of lecturers. Each student $s_i \in S$ has a preference list, ranking a subset $P_s i \subseteq P$ strictly. The set A_i is the set of projects which appear on student s_i his preference list. Each lecturer l_k teaches a set of projects $P_l k \subseteq P$. In case of the two-sided matching version, each lecturer l_i also has a preference ranking over the students which find his projects acceptable $\{s_i \in S : P_l k \cap A_i \neq \emptyset\}$. This SPA problem can be easily translated to a MPT scheduling problem by letting the lectures represent the resources, and letting each resource represent himself as a project, as well as letting combinations of resources be represented as projects.

The maximum capacity of both the projects as well as the lecturers should be set to 1.

Abraham et al. (2007) proposes an algorithm inspired by the Gale-Shapley algorithm described above which is proven to be strategy-proof and Pareto optimal. In this algorithm students apply to projects. These applications lead to provisional assignments for students on projects and lecturers. Each provisional assignment can be broken later in the execution.

Algorithm 1.2: SPA with provisional assignments

```
1  Let S be a set of students
2  Let ≺_s be an students' preference order over the projects
3  Let P be a set of projects
4  Let L be a set of lecturers
5  Let p_l be the lecturer assigned to a project
6  Let ≺_l be an lecturers' preference order over the students
7  Let π be a priority order over the students
8
9  For s ∈ S in order π do:
10   Assign s to project p =≺_s (S)
11   remove s from S
12   if p_l is over-subscribed || if p is over-subscribed:
13     Break assignment of project with lowest ≺_l (p_l)
14     re-add s to S
```

While the SPA problem extension and the existing tenants extension have solved two properties of MPT scheduling in isolation, a combination of the two is needed for these allocation methods to be applicable to MPT scheduling. No cases were found in the literature which attempt to solve a SPA problem with existing tenants, although it is easy to modify the algorithm to support this case. The algorithm proposed to solve this combined problem is a modification of the YGMH-IGYT Algorithm by Abdulkadiroglu and Sonmez (1999) and is explained in Algorithm 1.3.

This SPA implementation of the YGMH-IGYT algorithm reduces to a top trading cycle algorithm and is therefore strategy-proof and Pareto optimal.

Theorem 1. *For a given ordering f, the YRMP-IGYT algorithm has the same outcome as a top trading cycles algorithm*

Proof (after Abdulkadiroglu and Sonmez (1999, p. 251)). For any set A of agents, and P of projects remaining in the algorithm, there are two ways for the YRMP-IGYT algorithm to allocate the next series of projects:

Case 1: There is a cycle of agents $(a_1, ...a_k)$. In such a case, agent a_1 is assigned the project of or blocked by a_2, a_2 the project of or blocked by a_3 and a_k the project of or blocked by a_1. $(p_{a_1}, a_1, p_{a_2}, a_2, ...p_{a_k}, a_k)$ is a top trading cycle.

Case 2: There is a sequence of agents $(a_1, ...a_k)$ ordered by priority where a_1 demands the project of or blocked by a_2, a_2 the project of or blocked by a_3, a_k an available project p. $(p, a_1, p_{a_2}, a_2, ..., p_{a_k}, a_k)$ is then a top trading cycle. ■

Algorithm 1.3: You request my project, I get your turn

```
Let A be a set of agents
Let ≺ be an agents' preference order over the projects
Let P be a set of projects
Let L be a set of lecturers
Let π be a priority order over the agents

For a ∈ A in order π do:
Allocate ≺ (a) to a
if ≺ (a) is allocated to agent i && π(i) > π(a), do:
set π(i) = 0
end if

If a cycle forms C ⊆ A| ≺ (c1) = p_{c2}, ≺ (c2) = p_{c3}, ≺ (cn) = p_{c1}, , do:
Assign houses in trading circle
end if
```

In conclusion we can state that the SPA problem with existing tenants is capable of modelling the scheduling problem as present in Multiprocessor task problems. Furthermore, the agents themselves can safeguard the various properties for the specific FMPT, JMPT, and OMPT problem instances described.

The mechanism described above allows for several of the job characteristics as described in the comparison framework. Firstly, precedence relations between tasks can be safeguarded by the individual agents by removing incompatible machines from their preference lists. Release dates are automatically added by letting agents arrive and depart over time. Job characteristics which cannot be modelled are pre-emption, since it is not guaranteed that an agent will be able to reclaim previously used processors after releasing their claim on it. Furthermore, hard deadlines cannot be modelled.

It is clear that the mechanism allows for incorporating job preferences over machines. Machine preferences currently cannot be modelled as this requires a provisional assignment algorithm which can deal with existing tenants. Also, macro-level preferences over all the agents or machines cannot be modelled.

7 Combinatorial Auctions & Auction-based coordination mechanisms

The third and final solution strategy which will be discussed in this paper are Combinatorial auctions (CA), which have often played a key in MAS (Wellman, Wurman, Walsh, & MacKie-Mason, 1998; Kutanoglu & David Wu, 1999; Hunsberger & Grosz, 2000; Collins, Gini, & Mobasher, 2002; Amir, Sharon, & Stern, 2015) and seem to be a general approach for resource allocation among self-interested agents (Shoham & Leyton-Brown, 2009, p. 361). Combinatorial auctions are based on auction theory but instead of the agents bidding on single items, they bid on bundles of items. This technique has clear advantages in cases

where items are complementary in value, such as in for example radio spectrum auctions, and also multiprocessor tasks.

A CA is characterized by a set of bidders $N = \{1, ..., n\}$, and set of items $M = \{1, ..., m\}$. Bidders are interested in obtaining one or more bundles $S \subseteq M$ and the value of a bid by agent i on bundle S is denoted by $v_i(S)$. An allocation is described by $x_i(S) \in \{0, 1\}$, where $x_i(S) = 1$ if bundle S is allocated to agent i. An allocation is feasible if every item is allocated only once (Cramton et al., 2006, p. 557):

$$\sum_{i \in N} \sum_{S \subseteq M, S \ni j} x_i(S) \in \{0, 1\} \text{ for all } j \in M \tag{1}$$

When exploring CA mechanisms, there are three main design dimensions:

1. *Bidding rules:* How are bids made;
2. *Clearing rules:* How is the final allocation made based on the submitted bids;
3. *Information rules:* Who has what information in which stage of the auction.

CA provide a natural fit for MPT problems in the following way. There are two main types of agent: clients and an auctioneer. The clients submit bids on (bundles of) machines to the auctioneer, the auctioneer than takes this bids and attempts to find an allocation of items that maximizes his revenue. CA provide a natural fit on the multiprocessor task part of the scheduling problems since the parallel processors can be modelled as bundles in the auction mechanism. As with the HAP and SMP, CA are typically a single-time allocation method, but can easily be extended to incorporate a time dimension. For allowing scheduling over time, the bidding language of the auction should be altered to allow bids on bundles for specific time periods. This can either be a discretized time $T = \{0, 1, 2, t\}$ in which all items are added for all timeslots $I \times T$ are added to the auction, or a continuous time where the agents indicate precisely when they need a resource.

The clearing rules of the CA mechanism mostly deal with ensuring truthfulness and strategy-proofness of the auction. Making CA strategy-proof and Pareto optimal is a major topic of research and requires careful drafting of the used auction mechanism. Several CA mechanisms exist which can ensure truthfulness, such as the Vickrey-Clarke-Groves auction (Vickrey, 1961), iterative auctions (Parkes, 1999), and ascending proxy auctions (Ausubel, Cramton, & Milgrom, 2006; Ausubel & Milgrom, 2006). These auction mechanisms share the aspect that a payment is required from the agents to the auctioneer to ensure truthfulness, either using real currency or virtual currency (Ng, 2011; Vidal, Pla, Guijarro, & Martinez-Bauset, 2013). However, as Schummer and Vohra (2007, p. 233) note, "There are many important environments where money cannot be used as a medium of compensation", for example in organ donations because of ethical and legal restrictions (Ashlagi, Fischer, Kash, & Procaccia, 2010). In such cases, it is possible to determine the applicability of truthful CA mechanisms in absence of payments (Krysta & Ventre, 2010; Fotakis, Krysta, & Ventre, 2013),

although this greatly reduces the possible design space. Another option is to release the truthfulness and strategy-proofness constraints and instead, use CA as an auction-based coordination mechanism (Koenig, Keskinocak, & Tovey, 2010; Brouns, 2015).

CA is capable of modelling several of the job characteristics as described in the comparison framework. The pre-emption of jobs can be modelled by letting the agents bid on a bundle of items twice, with different time periods. Both precedence relations and release dates should be handled by the agents themselves, as they should not bid on machines they cannot use yet. Set-up times and hard deadlines are not possible to model.

The modelling of job preferences over machines is obvious by letting the agents vary their bid value on the machine bundles. Machine and macro-level preferences is possible but this can impact the strategy-proofness of the auction. For more information on this subject the reader is directed to (Engel, 2008; Loertscher & Marx, 2014).

In conclusion we can state that Combinatorial Auctions are well equipped to model the scheduling problem presented in multiprocessor task environments. Strategy-proofness and truthfulness can most easily be ensured by incorporating payments, although several options are available for truthful auctions without payments. Furthermore, in environments where strategy-proofness and truthfulness are not required, CA can be used effectively as coordination mechanisms.

8 Conclusion

The aim of this paper was to determine the applicability of three different Multi-agent System (MAS) architectures on scheduling problems with Multiprocessor tasks (MPT). The architectures which have been discussed in this article are Problem Decomposition, House Allocation and Stable Marriage problem, and Combinatorial Auctions. These have been compared based on three aspects: Their applicability to represent the MPT problem, their capacity to model various job characteristics, and their flexibility in representing agent preferences.

It was found that problem decomposition could represent the problem situation of MPT problems, but is unclear how much benefit is gained since MPT problems typically have higher dependencies between processors than regular job-shop problems. Furthermore, agents in problem decomposition are different from the agents meant in this article as they are a type of computational helpers, rather than true agents with preference orders over the allocation.

By extending the House Allocation and Stable Marriage problems to the House Allocation Problem with existing tenants and the Student/Project Allocation problem, it was possible to derive a strategy-proof and Pareto optimal algorithm for the allocation of multiprocessor tasks on a set of resources. This algorithm is based on the You Get My House - I Get Your Turn algorithm for the house allocation problem with existing tenants.

Combinatorial Auctions provide a good fit on Multiprocessor task scheduling as the agents are by default able to express their desire to obtain bundles of items.

A simple extension on the bidding language made it possible to incorporate a time domain in the auction mechanism. Combinatorial Auctions can be made strategy-proof and Pareto optimal by carefully designing the bidding, clearing, and information rules. When it is not possible to incorporate payments in the auction mechanism, the design space for strategy-proofness is drastically reduced although several options remain.

References

Abdulkadiroglu, A. & Sonmez, T. (1999). House Allocation with Existing Tenants. *Journal of Economic Theory*, *88*(December), 233–260.

Abraham, D. J., Irving, R. W., & Manlove, D. F. (2007). Two algorithms for the Student-Project Allocation problem. *Journal of Discrete Algorithms*, *5*(1), 73–90. doi:10.1016/j.jda.2006.03.006

Agnetis, A., Billaut, J.-C., Gawiejnowicz, S., Pacciarelli, D., & Soukhal, A. (2014). *Multiagent Scheduling*. Berlin Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-41880-8

Amir, O., Sharon, G., & Stern, R. (2015). Multi-Agent Pathfinding as a Combinatorial Auction. In *Twenty-ninth aaai conference on artificial intelligence (aaai-15)*. Austin, TX.

Ashlagi, I., Fischer, F., Kash, I. a., & Procaccia, A. D. (2010). Mix and Match. In *Proceedings of the 11th acm conference on electronic commerce* (pp. 305–314). New York: ACM.

Ausubel, L. M., Cramton, P., & Milgrom, P. (2006). The Clock-Proxy Auction: A Practical Combinatorial Auction Design. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (pp. 212–259). Cambridge, MA, USA: MIT Press.

Ausubel, L. M. & Milgrom, P. (2006). Ascending Proxy Auctions. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (pp. 150–212). Cambridge, MA, USA: MIT Press.

Brouns, M. (2015). *Operational Scheduling in a Multi-Actor Environment using Multiagent Systems - A case study on Multiprocessor Task Problems in Liquid Bulk Terminals* (MSc thesis, Delft University of Technology).

Brucker, P. (2007). *Scheduling Algorithms*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-69516-5

Brucker, P. & Krmer, A. (1995). Shop scheduling problems with multiprocessor tasks on dedicated processors. *Annals of Operations Research*, *57*(1), 13–27.

Collins, J. E., Gini, M., & Mobasher, B. (2002). Multi-agent negotiation using combinatorial auctions with precedence constraints. *University of Minnesota, Department of Computer Science and Engineering*, 1–34.

Cramton, P., Shoham, Y., & Steinberg, R. (2006). *Combinatorial auctions*. Cambridge, MA, USA: MIT Press. doi:10.7551/mitpress/9780262033428.001.0001

Engel, Y. (2008). Structured Preference Representation and Multiattribute Auctions.

Ergin, H. . (2000, August). Consistency in house allocation problems. *Journal of Mathematical Economics*, *34*(1), 77–97. doi:10.1016/S0304-4068(99)00038-5

Fotakis, D., Krysta, P., & Ventre, C. (2013). Combinatorial Auctions without Money. In *Proceedings of the 2014 international conference on autonomous agents and multi-agent systems* (pp. 1029–1036). arXiv: arXiv:1310.0177v1

Gabel, T. & Riedmiller, M. (2008). Adaptive Reactive Job-Shop Scheduling With Reinforcement Learning Agents. *International Journal of Information Technology and Intelligent Computing*.

Gale, D. & Shapley, L. S. (1962). College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, *69*(1), 9–15.

Gale, D. & Shapley, L. S. (962). College Admissions and the Stability of Marriage. *1The American Mathematical Monthly*, *69*(1), 9–15.

Graham, R., Lawler, E., Lenstra, J. K., & Kan, A. (1979). Sequencing and Scheduling: Algorithms and Complexity. In *Annals of discrete mathematics* (Chap. 9, Vol. 5, pp. 287–326). Annals of Discrete Mathematics. Elsevier. doi:10.1016/S0167-5060(08)70356-X

Hunsberger, L. & Grosz, B. (2000). A combinatorial auction for collaborative planning. In *Proceedings fourth international conference on multiagent systems*. doi:10.1109/ICMAS.2000.858447

Iwama, K., Miyazaki, S., & Yanagisawa, H. (2012). Improved approximation bounds for the Student-Project Allocation problem with preferences over projects. *Journal of Discrete Algorithms*, *13*, 59–66. doi:10.1016/j.jda.2012.02.001

Jalilzadeh, B., Planken, L., & De Weerdt, M. (2010). Mechanism design for the online allocation of items without monetary payments. *Lecture Notes in Business Information Processing*, *59*, 74–87. doi:10.1007/978-3-642-15117-0_6

Jansen, K. & Porkolab, L. (2000). for General Multiprocessor Job Shop Scheduling, 878–889.

Jennings, N. & Bussmann, S. (2003). Agent-based control systems. *IEEE Control Systems Magazine*, *23*(1), 61–74.

Koenig, S., Keskinocak, P., & Tovey, C. (2010). Progress on agent coordination with cooperative auctions. In *Twenty-fourth aaai conference on artificial intelligence*.

Kouider, A. & Bouzouia, B. (2012, January). Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation. *International Journal of Production Research*, *50*(2), 409–424. doi:10.1080/00207543.2010.539276

Kramer, A. (1995). *Scheduling preemptive multiprocessor tasks on dedicated processors* (Doctoral dissertation). doi:10.1016/0166-5316(94)90058-2

Krysta, P. & Ventre, C. (2010). Combinatorial auctions with verification are tractable. *Lecture Notes in Computer Science (including subseries Lecture*

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *6347 LNCS*(PART 2), 39–50. doi:10.1007/978-3-642-15781-3_4

Kutanoglu, E. & David Wu, S. (1999). On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. *IIE Transactions*, *31*(1), 813–826. doi:10.1080/07408179908969883

Kwanashie, A., Irving, R. W., Manlove, D. F., & Sng, C. T. S. (2014). Profile-based optimal matchings in the Student/Project Allocation problem. In *Proceedings of iwoca 2014: the 25th international workshop on combinatorial algorithms* (pp. 213–225). arXiv: 1403.0751

Lee, J. & Kim, C.-O. (2008). Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. *International Journal of Production Research*, *46*(February 2015), 233–265. doi:10.1080/00207540701441921

Liu, N., Abdelrahman, M. a., & Ramaswamy, S. (2005). Robust and adaptable job shop scheduling using multiple agents. In *Proceedings of the thirty-seventh southeastern symposium on system theory, 2005. ssst '05.* (pp. 77–81). Cookeville, TN: Ieee. doi:10.1109/SSST.2005.1460875

Loertscher, S. & Marx, L. M. (2014). *Can Auctioneers Effectively Favor Selected Bidders ? The Impact of Resale on Auctions with Bid Credits.*

Ng, C. (2011). *Online Mechanism and Virtual Currency Design for Distributed Systems* (Doctoral dissertation, Harvard University).

Parkes, D. C. (1999). iBundle: An Efficient Ascending Price Bundle Auction. In *In acm conference on electronic commerce (ec-99)* (pp. 148–157).

Pinedo, M. L. (2012). *Scheduling : Theory, Algorithms, and Systems.*

Qing-song, L. Q.-s. L. & Li-ming, D. L.-m. D. (2009). Model Design of Job Shop Scheduling Based on Multi-agent System. *2009 IITA International Conference on Services Science, Management and Engineering*, (1), 233–236. doi:10.1109/SSME.2009.117

Russel, S. & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd) (M. Hirsch, T. Dunkelberger, & M. Haggerty, Eds.). Upper Saddle River: Pearson Education.

Schummer, J. & Vohra, R. (2007). Mechanism design without money. In N. Nisan, T. Roughgarden, E. Tardos, & V. Vazirani (Eds.), *Algorithmic game theory* (Chap. 10). New York, New York, USA: Cambridge University Press.

Shoham, Y. & Leyton-Brown, K. (2009). *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations.* New York: Cambridge University Press.

Siekmann, J., Hartmanis, J., & Leeuwen, J. V. (2002). *Multi-Agent Systems and Applications II* (V. Mak, O. tpnkov, H. Krautwurmov, & M. Luck, Eds.). Springer Berlin Heidelberg. doi:10.1007/3-540-45982-0

Vickrey, W. (1961). Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, *16*(1), 8–37. doi:10.2307/2977633

Vidal, J. R., Pla, V., Guijarro, L., & Martinez-Bauset, J. (2013). Dynamic spectrum sharing in cognitive radio networks using truthful mechanisms and

virtual currency. *Ad Hoc Networks*, *11*(6), 1858–1873. doi:10.1016/j.adhoc.2013.04.010

Wang, Z. & Liu, Y. (2006). A multi-agent agile scheduling system for job-shop problem. In *Proceedings of the isda 2006: sixth international conference on intelligent systems design and applications* (Vol. 2, pp. 679–683). doi:10.1109/ISDA.2006.253918

Wellman, M. P., Wurman, P. R., Walsh, W. E., & MacKie-Mason, J. K. (1998). Auction Protocols for Decentralized Scheduling/Some Economics of Market-based Distributed Scheduling. *Eighteenth International Conference on Distributed Computing Systems*, (May), 1–31.

Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. Chichester, West Sussex: John Wiley & Sons Ltd.

Wu, Z. & Weng, M. X. (2005). Multiagent scheduling method with earliness and tardiness objectives in flexible job shops. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *35*(2), 293–301. doi:10.1109/TSMCB.2004.842412