ISET WORKING PAPER SERIES

WP 04-11

Task Assignment with Autonomous and Controlled Agents

Florian M. Biermann, Victor Naroditsky, Maria Polukarov, Alex Rogers, Nicholas R. Jennings

The International School of Economics at Tbilisi State University (ISET) is supported by BP, the Government of Georgia, the Norwegian Ministry of Foreign Affairs, Higher Education Support Program of the Open Society Institute, the Swedish International Development Agency and the World Bank.

International School of Economics at Tbilisi State University
16 Zandukeli Street, Tbilisi 0108, Georgia
www.iset.ge e-mail: publications@iset.ge

Task Assignment with Autonomous and Controlled Agents[*]

Florian M. Biermann[†] Victor Naroditskiy[‡] Maria Polukarov[‡] Alex Rogers[‡] Nicholas R. Jennings[‡]

Abstract: We analyse assignment problems in which not all agents are controlled by the central planner. The *autonomous* agents search for vacant tasks guided by their own preference orders defined over subsets of the available tasks. The goal of the central planner is to maximise the total value of the assignment, taking into account the behaviour of the uncontrolled agents. This setting can be found in numerous real-world situations, ranging from organisational economics to "crowdsourcing" and disaster response. We introduce the *Disjunctively Constrained Knapsack Game* and show that its unique Nash equilibrium reveals the optimal assignment for the controlled agents. This result allows us to find the solution of the problem using mathematical programming techniques.

1. Introduction

Problems in economic theory are traditionally analysed in terms of stable outcomes (equilibria) or efficient solutions (optima). In the former case, the problem is considered in the context of the interaction of rational, self-interested, autonomous agents; in the latter, the agents are assumed to follow the instructions of the central planner who aims to optimise some global objective. However, in realistic economic systems autonomous agents are often placed together with those controlled by the central planner; indeed, public and private sectors often interact in jointly tackling social problems or locating economic activities. Typically, the autonomous agents will act to obtain their own individual goals, and the problem of the central planner is to coordinate the controlled agents so as to optimise the overall performance of the system, while taking into account the behaviour of self-motivated participants. The present paper is offered with the motivation to investigate such "semi-autonomous" scenarios; Section 2 outlines multiple real-life situations which fit into this framework.

Specifically, we look at assignment problems where agents must be matched with indivisible tasks (or goods). Some players are autonomous, and face private

[*]The authors thank Ulrich Pferschy (University of Graz) and Takeo Yamada (National Defense Academy of Japan) for useful information and comments regarding *Disjunctively Constrained Knapsack Problems*, and Tobias Achterberg (IBM) for explanations regarding bilevel programming.

[†]International School of Economics at Tbilisi State University, Georgia. Email: f.biermann@iset.ge

[‡]School of Electronics and Computer Science, University of Southampton, UK. Emails (in order of authors): vn@ecs.soton.ac.uk, mp3@ecs.soton.ac.uk, acr@ecs.soton.ac.uk, nrj@ecs.soton.ac.uk

incentives to solve certain tasks; instead of submitting to the planner's will, these agents strive to obtain the task that rates most highly according to their own preference rankings. For technical simplicity we assume that the central planner (CP), who aims to maximise the value of the overall assignment (including the contributions of the autonomous agents), assigns the controlled agents *first*. The autonomous agents can only choose tasks which are left vacant by the CP. At the end of this section will be explained why this is indeed a weak assumption. We call this variation of the assignment problem *Semi-Autonomous Assignment Problem* (SAAP).

When all agents are controlled, the SAAP turns into a classical assignment problem.[1] In the classical assignment problem, an "assignment matrix" specifies the value obtained from each possible task-agent pair. The highest-valued assignment can for example be found with the *Hungarian Method* of Kuhn (1955).

The optimal solution of an SAAP from the point of view of the central planner corresponds to a stable matching (Gale and Shapley, 1962) in a particular *marriage market*[2] formed by autonomous agents and tasks. In this market, the preferences of the autonomous agents are their rankings over tasks, while the values of the "assignment matrix" determine the preferences of the tasks. By assigning the controlled agents, the central planner can block some tasks and in this way essentially determine the market in which the stable matching is formed.

Note that the autonomous agents, amended to the classical assignment problem, are assumed to have ordinal preferences over the available tasks. This considerably increases the robustness and applicability of our model. We neither require the central planner to form a belief about cardinal utility functions of the autonomous agents, nor do we assume the autonomous agents to be von Neumann-Morgenstern expected utility maximisers.[3] Likewise, adopting ordinal preferences allows us to directly utilise results from a branch of game theory, usually called *matching theory*, which originated with the seminal paper of Gale and Shapley (1962).[4] From the start, matching theory evolved without drawing on the theory of expected utility offered by Neumann and Morgenstern (1944).

Finally, the seemingly strong assumption that the central planner moves first does not affect the generality of our model. Making use of a result from matching literature, on page 7 we argue that the *order* in which the autonomous agents and the central planner make their moves does not affect the final assignment.

[1]Apparently, the classical assignment problem was defined first in a posthumously published paper by Carl Gustav Jacob Jacobi, originating from the first half of the 19th century. Moreover, Jacobi's article (written in Latin) may have already antedated the *Hungarian Method* of Kuhn (1955). For an investigation into the history of assignment problems and further literature references, see Martello (2010).

[2]Technically, a marriage market is a one-to-one two-sided matching problem.

[3]In our model, the autonomous agents are not farsighted, which is a reasonable assumptions in many of the applications we have in mind (cf. Section 2). In principle, their behaviour could be governed by simple decision heuristics as put forward by many psychologists (e.g. Simon (1957), Gigerenzer and Todd (2000)), as long as these heuristics give rise to linear orderings of the alternatives (which is often not the case).

[4]Game-theoretic matching theory is not the same as graph-theoretic matching theory. The classical survey of game-theoretic matching theory is Roth and Sotomayor (1990).

This is true as long as it is at the disposal of the central planner to replace autonomous agents who occupy tasks by controlled agents, an assumption not too unrealistic for those applications we describe in Section 2. If two autonomous agents compete for one vacant task, the task is granted to the agent who is better at performing the task.

The rest of the paper is organised as follows. Section 2 motivates our work by describing several real-life situations which resemble SAAPs. The model is then formally defined in Section 3. Section 4 introduces *Disjunctively Constrained Knapsack Games* (DCKG) and proves that the unique Nash equilibrium of a DCKG corresponds to an optimal assignment of an underlying SAAP. In Section 5, we show that the Nash equilibrium of a DCKG can be characterised as the solution of a bilevel mathematical program. We conclude in Section 6 with directions for future work.

2. Real-world examples

Semi-autonomous assignment problems arise naturally in the context of *location of economic activities*. In Koopmans and Beckmann (1957), for example, the authors discuss the assignment problem in the context of choosing locations for industrial plants under the standard assumption that the central planner is responsible for choosing location for all of the plants. However, in reality such tasks are typically divided between the public and the private sectors, where private businesses strive to maximise their own profits and the government is concerned with the overall welfare of the society. Note also that state institutions often have the priority over private entrepreneurs in making their choices, consistent with the assumptions of our model.

As another example, consider *private-public partnerships* (PPP), where the public party, which usually supervises the complete project, intends to advance some public goal. In contrast, the participating private parties are primarily interested in those subprojects which have commercial potential. This poses an obstacle for assigning tasks in a globally optimal way. Companies will try to avoid those tasks which are unprofitable and difficult, trying instead to obtain subprojects promising high profits at low risk. A typical example is the provision of health care through hospitals and doctors, which is facilitated through private-public partnerships in many countries.[5] The payment agreements between the government and the private partners usually do not reimburse a hospital or doctor for *exactly* those costs associated with a specific patient. As a result, patients (= "tasks") yield different profit opportunities. Although hospitals/doctors (= "agents") participating in a PPP are not formally entitled to pick the profitable patients and reject the others, there may be informal ways to deter unprofitable

[5]For an overview of private-public partnerships in the health sector, see Nikolic and Maikisch (2006).

patients.[6] The model presented in this paper could thereby prescribe an optimal policy for a public health system which both directly employs medical resources (doctors, hospitals etc.) and engages private contractors.

In the Internet economy, the so-called *crowdsourcing systems* (see, e.g., Benkler (2006); Brabham (2008); Howe (2008)) can also be modelled with SAAPs. In a crowdsourcing system, tasks which cannot satisfactorily be solved without human expertise are assigned to a group of more or less anonymous amateur problem solvers (the "crowd"). Yet companies making use of crowdsourcing do not have to totally rely on the crowd. For some of the tasks or even for all of them, they can engage professional problem solvers. These belong to their own personnel or a contractor's personnel who cannot reject tasks assigned to them. In contrast, crowd members can freely choose which tasks to work on, and they are probably not indifferent between all tasks. Hence, the firm has to find an optimal way of distributing its tasks between professional and amateur problem solvers.

Disaster response situations providing prominent examples of crowdsourcing can also be analysed with our model. Consider a disaster relief situation where professional disaster responders coordinated by the government are assisted by local residents and disaster survivors. The government has neither the communication capabilities nor the authority to tell local participants what to do. However, local participants are very helpful and their efforts should not be ignored. Assuming the government can estimate the preferences of local participants (e.g., they visit sites in order of distance from their home), our work provides a way for the government to assign professional disaster responders optimally.

Finally, autonomous task choice can even be observed in military organisations, which are famous for their strict adherence to the principle of obeying orders.[7] If solving critical tasks is "prestigious" in some sense, there may be an incentive for players to unilaterally go for those critical tasks, disregarding the assignment the central planner would prefer. In military history it regularly occurred that ambitious commanders tried to gain fame by acting more bravely or by taking greater risks than desired by the central command. An outstanding example is the celebrated Danish naval officer Peter Jansen Wessel (1691-1720), called *Tordenskjold* (Danish for "thunder shield"). He constantly strived for the most prestigious tasks in the *Great Northern War* (1700-1721),

[6]By entering "hospital turns away" or a similar phrase into an internet search engine, one gets plenty of media reports about exactly this issue. For example, UK dentists, working for the *National Health Service*, arguably behaved in such a way (Templeton, 2007). Reports about hospitals being reluctant to examine patients with X-ray or brain scans may straightforwardly be interpreted as avoidance of unprofitable tasks.

[7]Situations resembling SAAPs can be found not only *within* military organisations. The 2011 war in Libya was fought by a coalition of NATO and loosely organised rebel troops who jointly tried to overthrow the regime of dictator Muammar Gaddafi. While the NATO forces were totally coordinated, it was arguably difficult to coordinate the actions of the rebels, who were untrained, unprofessional, and lacked command chains. Consequently, the NATO, as the central planner of the SAAP, had to anticipate the prospective actions of the rebels when making its decisions on air strikes.

thereby notoriously disobeying orders.[8] His confrontation with the Swedish fleet in the *Battle of Dynekilen* (1716) in which his 7 ships captured 31 Swedish ships and destroyed another 13, was not backed by orders of the admiralty.[9] Wessel's anarchistic conduct evoked considerable criticism in the Danish admiralty, eventually leading to a trial at a court-martial. Yet he was acquitted and even made an admiral later.[10] His disobedience yielded huge personal prestige, as can be seen from the fact that Wessel is praised in the national anthems of both Denmark and Norway (the country he originated from).

3. Semi-Autonomous Assignment Problems

Before formally defining our model, let us first recall the definition of a classical *Assignment Problem* (AP). An AP is defined by a triple (A, T, v), where A is a set of *agents*, T is a set of *tasks*, and v is an *evaluation function* which maps $A \times T$ into $\mathbb{R}_+ \cup \{0\}$. The problem is to find an *assignment* (or, *matching*) of agents to tasks for which the sum of the values of pairs matched is maximised. Formally, an assignment μ is a subset of $A \times T$ such that no two distinct pairs in μ share a player or a task, that is:

$$(a,t),(\hat{a},\hat{t}) \in \mu : (a,t) \neq (\hat{a},\hat{t}) \Rightarrow a \neq \hat{a} \ \wedge \ t \neq \hat{t}.$$

The objective of the central planner is then given by

$$\max_{\mu \in \boldsymbol{\mu}} \sum_{(a,t)\in\mu} v((a,t)),$$

with $\boldsymbol{\mu}$ being the set of all assignments which can be formed from the set $A \times T$.

We now generalise the AP model to what we call the *Semi-Autonomous Assignment Problem* (SAAP). An SAAP is defined by a tuple

$$(C \cup F, T, v, \succ_{\boldsymbol{F}}),$$

where C and F are two disjoint sets, and we set $A := C \cup F$. As before, we refer to the elements of A as *agents* (or, *players*), while the members of C are termed *coordinated* (or, *controlled*) and the members of F are referred to as *free* (or, *autonomous*). The function v is defined as before, and $\succ_{\boldsymbol{F}}$ is a *preference profile* which contains, for each free agent $f \in F$, a linear preference order[11] $\succ_f$ defined over a set $\mathcal{T}_f \subseteq T$. The tasks in $\mathcal{T}_f$ are interpreted to be those which can be accomplished by f. Given this, the central planner of an SAAP aims to find

$$\max_{\mu \in \boldsymbol{\mu}^{\text{SAAP}}} \sum_{(a,t)\in\mu} v((a,t)),$$

[8] For an account of his deeds, see Chapter 1 ("A Knight Errant of the Seas") in Riis (2007).

[9] "He could not go back and ask for permission, and one may shrewdly guess that he did not want to, for it would certainly have been refused." (Riis (2007), p. 10).

[10] Cf. Riis (2007), pp. 6 and 9.

[11] A linear ordering is irreflexive, asymmetric, and transitive.

where $\boldsymbol{\mu}^{\mathrm{SAAP}}$ is the set of *SAAP-feasible* assignments defined next.

First, we have to specify the behaviour of the free players, i.e. the way in which they allocate themselves to tasks. For simplicity, we assume ties do not arise:

$$(a,t),(\hat{a},\hat{t}) \in A \times T: \quad (a,t) \neq (\hat{a},\hat{t}) \Rightarrow v((a,t)) \neq v((\hat{a},\hat{t})) \tag{3.1}$$

and

$$\mu,\mu' \in \boldsymbol{\mu} : \mu \neq \mu' \Rightarrow v(\mu) \neq v(\mu'), \tag{3.2}$$

with $v(\mu) := \sum_{(a,t)\in\mu} v((a,t))$.

The search process of the free players proceeds as follows: after the coordinated agents were assigned to tasks by the central planner, each free agent f approaches the task $t := \max_{\succ_f} \mathcal{T}_f$. If f finds t to be vacant, f takes over t. If f finds that a coordinated player already occupies t, f proceeds to the task which is second according to the preferences $\succ_f$, namely $t' := \max_{\succ_f} \mathcal{T}_f \setminus \{t\}$. Again, f checks the availability of t' and either takes it or continues with the subsequent item in its priority list. If there are no tasks left on f's priority list which were not yet approached, f stays idle.

For two free players f' and f'' it may be the case that $\mathcal{T}_{f'} \cap \mathcal{T}_{f''} \neq \emptyset$. So what happens if f' and f'' approach the same task t? In this case, we assume that the agent preferred by the task (i.e., better at performing the task)

$$\arg\max_{a\in\{f',f''\}} v((a,t))$$

keeps to t, while the other free agent continues the search process. This is a realistic assumption for scenarios in which free players, though being uncoordinated, have an interest in a high-valued solution of the problem (like in the disaster response application outlined in Section 2). However, our optimality result could be adjusted if another tie-breaking rule would be used instead.

We call this procedure the *Deferred Acceptance Algorithm with Blocked Tasks* (DAB). In the appendix we include a formal description of the algorithm and prove that it is finite and produces a unique output (see page 16). We now turn to discuss some of its important properties.

Consider the *Deferred Acceptance Algorithm* of Gale and Shapley (1962) that constructs a stable matching in a *marriage market*. A marriage market is defined as a triple $(M, W, \succ)$, where M is the set of "men" and W is the set of "women". A preference profile $\succ$ maps each $m \in M$ into a linear preference order defined over $W \cup \{m\}$, and each $w \in W$ into a linear preference order defined over $M \cup \{w\}$ (the item x in x's preference order stands for the option of being single).[12] In terms of SAAP, let the tasks stand for the women and the free agents stand for the men. Then, the DAB search process coincides with the Deferred Acceptance Algorithm of Gale and Shapley (1962), executed in a restricted marriage market which does not include the controlled agents and the tasks they occupy. In this market, the preferences of the free agents are given by their preferences over subsets of tasks. All tasks which are not in the set $\mathcal{T}_f$ are

[12]For a comprehensive discussion of marriage markets see Roth and Sotomayor (1990), chapter 2.

considered unacceptable for f. The preferences of tasks over agents are given by the value function v: a task t prefers f' over f'' iff $v((f',t)) > v((f'',t))$. The definition of a stable matching in this context is postponed to the end of this section (as we do not need it for the definition of the set $\boldsymbol{\mu}^{\text{SAAP}}$).

The order in which the free players propose to tasks, and the order in which they are rejected, does not influence the outcome assignment as the DAB procedure leads to a unique outcome. This was shown by McVitie and Wilson (1971), who modified the original algorithm of Gale and Shapley (1962) so as to let men propose to women sequentially and in an arbitrary order (in Gale and Shapley (1962), the men propose simultaneously at each stage). They proved that the matching resulting from their algorithm is identical to the one generated by the standard deferred acceptance algorithm. Remarkably, this finding of McVitie and Wilson (1971) also implies that the outcome of the DAB algorithm is not affected by our assumption that the central planner assigns the coordinated agents first; in the DAB search process, the output matching would be the same even if the CP would assign the controlled agents when the free agents were already searching in the market. This is true as long as the coordinated agents could take away any task already occupied by a free agent, an assumption we think is not too unrealistic for those applications we described in Section 2. However, for ease of exposition, we will keep our assumption.

We define a *coordinated assignment* to be a matching $\mu_C \subseteq C \times T$ (no free player f is a member of any pair in μ_C). Similarly, a *free assignment* is a matching $\mu_F \subseteq F \times T$ (no coordinated player c is a member of any pair in μ_F). We denote by $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$ a marriage market formed by free agents and those tasks which are not matched under μ_C. Formally,

$$(F, T, \succ_{\boldsymbol{F}})_{\mu_C} = (F, T \setminus \{t \mid (c,t) \in \mu_C\}, \succ_{\boldsymbol{F}}).$$

Given this, the set $\boldsymbol{\mu}^{\text{SAAP}}$ consists of the following assignments:

Definition 1. *An assignment μ is* SAAP-feasible *for a semi-autonomous assignment problem $(C \cup F, T, v, \succ_{\boldsymbol{F}})$ if $\mu = \mu_F \cup \mu_C$ and the matching μ_F is the outcome of the DAB algorithm in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$.*

Let μ_F be a free assignment in a marriage market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$. We say that μ_F is *stable* if it has no blocking pairs as defined below:

Definition 2. *A* blocking pair *for a free assignment μ_F in a marriage market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$ is a pair $(f,t) \in F \times (T \setminus \{t \mid (c,t) \in \mu_C\})$ with $t \in \mathcal{T}_f$, and* one *of the following four cases is fulfilled:*

1. *f is unmatched under μ_F, and t is matched to an agent f' with $v((f,t)) > v((f',t))$.*
2. *t is unmatched under μ_F and f is matched to a position t' with $t \succ_f t'$.*
3. *t and f are unmatched under μ_F.*
4. *f and t are matched to t' and f' respectively with $v((f,t)) > v((f',t))$ and $t \succ_f t'$.*

4. Disjunctively Constrained Knapsack Games

Given the above description of SAAP, we now characterise its solution as the unique Nash equilibrium of a two-player constant-sum game, termed the *Disjunctively Constrained Knapsack Game* (DCKG). This allows us to formalise the decision situation faced by the central planner as a max-min optimisation problem as in Neumann (1928). In the following section, we will then set up a mathematical program which identifies the unique Nash equilibrium of this game.

Besides being instrumental for deriving the program, the game representation of SAAP nicely reveals the problem's inner structure. Its formulation as a mathematical program then becomes natural and immediate. A similar approach has previously been used for a classical assignment problem. In Neumann (1953), a two-person zero-sum game was proposed where one player selects a field (a, t) in a checkerboard of n rows and n columns, and the other player guesses either the row a or the column t in which this field is found. It was shown that in an equilibrium the first player's strategy is a probability distribution that assigns positive probabilities to the pairs in an optimal assignment. We note however, that the DCKG model we present in this paper is rather different (and this difference is reflected in its mathematical programming formulation), in that it is more involved than the classical model.[13]

Consider the following game in extensive form, played between the CP and its adversary AD (a "Stackelberg game"). The CP and the AD have to jointly fill a knapsack. The CP moves first and chooses a subset μ_C of elements from the set $C \times T$ to be put into the knapsack. Afterwards, the AD chooses a set μ_F from the set $F \times T$ to be put into the knapsack, but AD must obey the restriction that μ_F is a stable matching in the market $(F, T, \succ)_{\mu_C}$. That is, the CP has the strategy set $S_{\text{CP}} = \boldsymbol{\mu_C}$, while the AD's strategy set is given by:

$$S_{\text{AD}} = \{s : \boldsymbol{\mu_C} \to \boldsymbol{\mu_F} \mid s(\mu_C) \text{ is a stable matching in } (F, T, \succ_{\boldsymbol{F}})_{\mu_C}\},$$

where $\boldsymbol{\mu_C}$ and $\boldsymbol{\mu_F}$ denote the sets of coordinated and free assignments of a given SAAP, respectively. Verbally, an element of S_{AD} is a function s which maps each choice $\mu_C \in \boldsymbol{\mu_C}$ of the CP into a reply $\mu_F := s(\mu_C) \in \boldsymbol{\mu_F}$ of the AD, and s must be such that μ_F is a stable matching in $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$.

The game is constant-sum. While the CP wants to *maximise* the value of the knapsack, which is given by

$$v(\mu_C \cup \mu_F) = \sum_{(a,t) \in \mu_C \cup \mu_F} v((a, t)),$$

the AD wants to *minimise* that value.[14]

We call this game a *Disjunctively Constrained Knapsack Game* (DCKG), because it has an obvious connection to the *Disjunctively Constrained Knapsack*

[13]See Section 5 for a detailed discussion.

[14]For example, set the payoffs of the players as $v(\mu_C \cup \mu_F)$ for the CP and $-v(\mu_C \cup \mu_F)$ for the AD. The resulting game is zero-sum.

Problem[15] (cf. Yamada et al. (2002)).[16] We show that the unique Nash equilibrium of this game determines the coordinated assignment μ_C^* which maximises the objective function of the corresponding SAAP.

Proposition 1. *A DCKG has a unique Nash equilibrium* (μ_C^*, μ_F^*)*. For* $\mu^* := \mu_C^* \cup \mu_F^*$ *holds* $\mu^* \in \boldsymbol{\mu}^{SAAP}$*, with* $\boldsymbol{\mu}^{SAAP}$ *being the set of matchings feasible for the underlying SAAP. Moreover,*

$$\mu^* \in \arg\max_{\mu \in \boldsymbol{\mu}^{SAAP}} \sum_{(a,t)\in\mu} v((a,t)).$$

Proof. The fact that a DCKG has a unique Nash equilibrium follows from condition (3.2) on page 6 and the fact that in a constant-sum game, all Nash equilibria yield the same payoff vectors.[17]

For proving the second statement, we first show that for any strategy μ_C played by the CP, the free matching generated by the DAB algorithm (on page 16) is a best response of the AD. Take μ_C as given, and let $\mu_F[\mu_C]$ denote the free assignment constructed by the DAB algorithm in the marriage market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$. As the DAB algorithm coincides with the Gale-Shapley algorithm in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$ (with free agents proposing), the matching $\mu_F[\mu_C]$ is the free agent optimal stable matching in the marriage market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$. This means that *all* free agents find $\mu_F[\mu_C]$ at least as good as any other stable matching in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$. It is a well-known result in matching theory that the optimal stable matching for the proposing side coincides with the *worst* stable matching of the responding side.[18] Hence $\mu_F[\mu_C]$ is the worst stable matching for the tasks, i.e. all tasks are at least as well off under any other stable matching in $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$. As any task t prefers f' over f'' iff $v((f', t)) > v((f'', t))$, this means that for any other stable matching $\hat{\mu}_F$ in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$ holds

$$\sum_{(f,t)\in\hat{\mu}_F} v((f,t)) > \sum_{(f,t)\in\mu_F[\mu_C]} v((f,t)).$$

Therefore $\mu_F[\mu_C]$, the matching constructed by the DAB algorithm, is a best reply to μ_C. As μ_C was chosen arbitrarily, the DAB algorithm constructs the best reply for *any* choice of the CP. So if μ_C^*, μ_F^* is the unique Nash Equilibrium of the DCKG, then $\mu_F^* = \mu_F[\mu_C^*]$. By Definition 1 on page 7 it holds that

[15] Also referred to as *Knapsack Problems with Conflict Graphs* (cf. Pferschy and Schauer (2009)).

[16] The decision problem of the CP could also be interpreted to be a *Max-Min 0-1 Knapsack Problem* (cf. Yu (1996)). In that interpretation, each of the AD's strategy choices would be one of the "scenarios" that the CP faces. Furthermore, the DCKG resembles the *Knapsack Sharing Problem (KSP)* as it was defined by Yamada and Futakawa (1997) (there is an earlier, less general definition by Brown (1979)). In both the KSP and the DCKG, different parties have to fill the knapsack. However, both models differ in many other aspects, for example in their objective functions.

[17] See, for example, Osborne and Rubinstein (1994), Proposition 22.2(b) on p. 22.

[18] See, for example, Knuth (1997), p. 13, or Roth and Sotomayor (1990), Theorem 2.13 and Corollary 2.14, p. 33.

$\mu_C \cup \mu_F[\mu_C] \in \boldsymbol{\mu}^{\text{SAAP}}$ for all coordinated matchings μ_C. Thus $\mu_C^* \cup \mu_F[\mu_C^*] \in \boldsymbol{\mu}^{\text{SAAP}}$, implying $\mu_C^* \cup \mu_F^* \in \boldsymbol{\mu}^{\text{SAAP}}$.

Finally, if there was another matching $\hat{\mu}_C$ with

$$\sum_{(a,t)\in\hat{\mu}_C\cup\mu_F[\hat{\mu}_C]} v((a,t)) > \sum_{(a,t)\in\mu_C^*\cup\mu_F[\mu_C^*]} v((a,t)),$$

then μ_C^* would be no best reply against the strategy of AD (a strategy of the AD specifies a reply matching μ_F for *every* coordinated matching chosen by the CP). CP could obtain a better payoff by switching from strategy μ_C^* to $\hat{\mu}_C$. As we assumed μ_C^*, μ_F^* to be a Nash equilibrium, this is impossible. Therefore $\mu^* := \mu_C^* \cup \mu_F^*$ solves the problem $\max_{\mu\in\boldsymbol{\mu}^{\text{SAAP}}} \sum_{(a,t)\in\mu} v((a,t))$. □

Now, for a zero-sum game with player set $\{1,2\}$, strategy sets S_1 and S_2, and payoff functions $p_1 := p(x \in S_1, y \in S_2)$ and $p_2 := -p(x \in S_1, y \in S_2)$, Neumann (1928) shows that player 1 has to solve the problem

$$\max_{x\in S_1} \min_{y\in S_2} p(x,y).$$

If we apply this general form to our problem, we get the formulation

$$\max_{\mu_C\in S_{\text{CP}}} \min_{\mu_F\in S_{\text{AD}}} \sum_{(a,t)\in\mu_C\cup\mu_F} v((a,t)).$$

Removing the definitions of the action sets from the objective function and taking care of them through constraints, we obtain:

$$\begin{aligned} &\max_{\mu_C\subseteq C\times T} \quad \min_{\mu_F\subseteq F\times T} \sum_{(a,t)\in\mu_C\cup\mu_F} v((a,t)) \\ &\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (4.1) \\ &\mu_C \cup \mu_F \text{ must be a matching.} \\ &\mu_F \text{ must be a stable matching in } (F, T, \succ_{\boldsymbol{F}})_{\mu_C}. \end{aligned}$$

The problem in (4.1) is stated informally. In the following section, we translate it into a form which can be tackled with mathematical programming techniques.

5. Mathematical program for computing the DCKG equilibrium

In this section, we formulate a mathematical program for finding the Nash equilibrium of a DCKG. An economic problem of a two-person Stackelberg game is a typical *bilevel* optimisation problem—a hierarchical program in which the set of constraints contains a parametric optimisation problem. In other words, the response of the lower level problem faced by the follower (in our case, AD) depends on the decision of upper-level problem of the leader (CP). Mathematically, this means that the set of decision variables can be divided into two parts

x and y, with x corresponding to the strategy of the upper level problem and y representing the response of the lower level problem parameterised in x.

Let us define the control variables. Set $\boldsymbol{\mathcal{M}} = T \cup A \cup (A \times T)$ and define for each $m \in \boldsymbol{\mathcal{M}}$ a variable $x(m) \in \{0, 1\}$. We denote the set of variables $(x(m))_{m \in \boldsymbol{\mathcal{M}}}$ by $\boldsymbol{x}$, and we denote a specific configuration[19] of the control variables (a "realization") by $\bar{\boldsymbol{x}}$. Each $\bar{\boldsymbol{x}}$ is associated with a set $\mu(\bar{\boldsymbol{x}}) \subseteq A \times T$ defined as

$$\mu(\bar{\boldsymbol{x}}) = \{(a, t) \in A \times T \mid \bar{x}((a, t)) = 1\}. \tag{5.1}$$

We now set up restrictions on $\boldsymbol{x}$ which make sure that for every possible value $\bar{\boldsymbol{x}}$ of $\boldsymbol{x}$ the set $\mu(\bar{\boldsymbol{x}})$ is indeed an *assignment*. For each $t \in T$, define a set

$$\mathcal{M}_t := \{t\} \cup \{(a, \hat{t}) \in A \times T \mid \hat{t} = t\}$$

which contains t and all pairs in $A \times T$ of which t is a member. Similarly, for each $a \in A$, define

$$\mathcal{N}_a := \{a\} \cup \{(\hat{a}, t) \in A \times T \mid \hat{a} = a\}$$

to be the set containing a and all pairs in $A \times T$ of which agent a is a member. The following two conditions on $\boldsymbol{x}$ ensure that for any configuration $\bar{\boldsymbol{x}}$ of $\boldsymbol{x}$, the set $\mu(\bar{\boldsymbol{x}})$ is an assignment:

$$\forall t \in T : \qquad \sum_{m \in \mathcal{M}_t} x(m) = 1 \tag{5.2}$$

$$\forall a \in A : \qquad \sum_{m \in \mathcal{N}_a} x(m) = 1 \tag{5.3}$$

These restrictions obviously prevent any task or any agent from being a member of more than one pair in $\mu(\bar{\boldsymbol{x}})$. Recall that $t \in \mathcal{M}_t$ and $a \in \mathcal{N}_a$. So (5.2) makes sure that for any configuration of variables $\bar{\boldsymbol{x}}$ holds $x(t) = 1$ iff in the matching $\mu(\bar{\boldsymbol{x}})$ the task t is not occupied by any agent. Likewise, by (5.3), for any configuration of variables $\bar{\boldsymbol{x}}$ it holds $x(a) = 1$ iff in the matching $\mu(\bar{\boldsymbol{x}})$ the agent a is unmatched. This will be important in what follows now.

We must ensure that for any $\bar{\boldsymbol{x}}$ the set $\mu(\bar{\boldsymbol{x}}) \cap (F \times T)$ is a stable matching in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu(\bar{\boldsymbol{x}}) \cap (C \times T)}$. To this end, we define four "conflict sets"[20] which correspond to the 4 cases in the definition of a blocking pair (Definition 2 on page 7). We will argue that an assignment $\mu(\bar{\boldsymbol{x}}) \cap (F \times T)$ is stable in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu(\bar{\boldsymbol{x}}) \cap (C \times T)}$ if in each of these conflict sets there is at most one

[19]Roth and Sotomayor (1990), p. 69, use the same terminology.

[20]We learned about this way to implement conflicts into a mathematical program through Yamada et al. (2002).

element m with $\bar{x}(m) = 1$.[21]

$$\begin{array}{ll} E_1 & := \{(f,(\tilde{f},\tilde{t})) \in F \times (F \times T) \ \mid \ v((f,\tilde{t})) > v((\tilde{f},\tilde{t}), \tilde{t} \in \mathcal{T}_f\} \\ E_2 & := \{(t,(\tilde{f},\tilde{t})) \in T \times (F \times T) \ \mid \ t \succ_{\tilde{f}} \tilde{t}\} \\ E_3 & := \{(t,f) \in T \times F \ \mid \ t \in \mathcal{T}_f\} \\ E_4 & := \{((f,t),(\tilde{f},\tilde{t})) \in (F \times T) \times (F \times T) \ \mid \ t \succ_{\tilde{f}} \tilde{t} \ \wedge \ v((\tilde{f},t)) > v((f,t))\}. \end{array}$$

To illustrate how these sets work, consider an element $(m', m'') \in E_2$. By definition of E_2, m' must be a task $\hat{t} \in T$. If in a configuration $\bar{x}$ it holds that $\bar{x}(m') = 1$, then by (5.2) for all pairs $(a,\hat{t}) \in \boldsymbol{\mathcal{M}}$ it must hold $\bar{x}((a,\hat{t})) = 0$. This means that $\hat{t}$ will be a vacant task in the matching $\mu(\bar{\boldsymbol{x}})$. If at the same time $\bar{x}(m'') = 1$, then by definition of E_2, a pair $m'' = (\tilde{f},\tilde{t})$ with $t \succ_{\tilde{f}} \tilde{t}$ is also in the matching $\mu(\bar{\boldsymbol{x}})$. Then according to case 2 in Definition 2, $(\tilde{f}, t)$ is a blocking pair against $\mu(\bar{\boldsymbol{x}}) \cap (F \times T)$ in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu(\bar{\boldsymbol{x}}) \cap (C \times T)}$ and $\tilde{f}$ would prefer the vacant position t over the position $\tilde{t}$ to which $\tilde{f}$ is assigned under $\mu(\bar{\boldsymbol{x}})$.

By analogous arguments, any elements $(m', m'') \in E_1$, $(m', m'') \in E_3$, and $(m', m'') \in E_4$ would lead to a blocking pair if $\bar{x}(m') = \bar{x}(m'') = 1$. To exclude these possibilities, we set $E := E_1 \cup E_2 \cup E_3 \cup E_4$ and demand

$$\forall (m', m'') \in E: \qquad x(m') + x(m'') \leq 1. \tag{5.4}$$

We have now established the following result:

Proposition 2. *A configuration $\bar{\boldsymbol{x}}$ of the variables $\boldsymbol{x}$ which fulfils the conditions (5.2),(5.3), and (5.4) generates a set $\mu(\bar{\boldsymbol{x}})$ with the following properties:*

1. *$\mu(\bar{\boldsymbol{x}})$ is an assignment (by (5.2) and (5.3)).*
2. *The free assignment $\mu(\bar{\boldsymbol{x}}) \cap (F \times T)$ is a stable matching in the marriage market $(F, T, \succ_{\boldsymbol{F}})_{\mu(\bar{\boldsymbol{x}}) \cap (C \times T)}$ (by (5.4)).*

The missing part in the mathematical program is the objective function. In the objective function of (4.1) there are two matchings μ_C and μ_F. The only thing we have to do is to translate these two matchings into a configuration of variables by rule (5.1). In this way, we straightforwardly obtain the objective function:

$$\max_{x(m): m \in \boldsymbol{\mathcal{M}} \cap (C \times T)} \ \min_{x(m): m \in \boldsymbol{\mathcal{M}} \cap (F \times T)} \ \sum_{m \in \boldsymbol{\mathcal{M}} \cap (A \times T)} x(m) \cdot v(m).$$

[21] One could also guarantee stability of $\mu(\bar{\boldsymbol{x}}) \cap (F \times T)$ in the market $(F, T, \succ_{\boldsymbol{F}})_{\mu(\bar{\boldsymbol{x}}) \cap (C \times T)}$ by restriction (3) of Vande Vate (1989), p. 148, or its generalisation of Rothblum (1992) (see Lemma 1 on page 59). This would allow for simply setting $\boldsymbol{\mathcal{M}} = A \times T$. However, the difference is mainly notational, as the formulation of Rothblum (1992) makes use of sets which are defined through the players' preferences, similar to our *conflict sets.* The notation $\sum_{j >_m w} x_{mj}$ in Rothblum (1992) denotes the summation over such a set, and his proof of Lemma 1 directly refers to the four cases we state in Definition 2. As we want to point out the connection between the mathematical program and Definition 2, we do not use the compact notation of these papers.

Putting everything together, the mathematical program which solves a SAAP by finding a Nash equilibrium of the associated DCKG is given by:

$$\begin{aligned}
\max_{x(m):m\in\boldsymbol{\mathcal{M}}\cap(C\times T)} \quad & \min_{x(m):m\in\boldsymbol{\mathcal{M}}\cap(F\times T)} \sum_{m\in\boldsymbol{\mathcal{M}}\cap(A\times T)} x(m)\cdot v(m) \\
\text{s.t.} \quad & \\
\forall m \in \boldsymbol{\mathcal{M}}: \quad & x(m)\in\{0,1\} \\
\forall t \in T: \quad & \sum_{m\in\mathcal{M}_t} x(m) = 1 \\
\forall a \in A: \quad & \sum_{m\in\mathcal{N}_a} x(m) = 1 \\
\forall (m',m'') \in E: \quad & x(m') + x(m'') \le 1
\end{aligned} \tag{5.5}$$

A decomposition of the program (5.5) into first-level maximisation and second-level minimisation can be found in Appendix B on page 18. For solution techniques in (discrete) bilevel optimisation, we refer the reader to the overview in Colson et al. (2007).

6. Conclusions

Our work introduces optimisation problems in which autonomous agents are placed together with those fully controlled by a central planner. The autonomous agents act to obtain their own individual goals. The central planner coordinates the controlled agents with the aim to optimise the overall performance of the system, while taking into account the behaviour of the self-motivated participants. This scenario is typical in realistic economic systems, some of which were outlined in Section 2. Specifically, we considered the Semi-Autonomous Assignment Problem (SAAP) in which the controlled agents are assigned by the central planner, while the free agents search for vacant tasks according to their own preference orders over subsets of accessible tasks.

Clearly, the search process assumed for the free agents is not the only reasonable way to put things. Indeed, there are many other possibilities for how one could model the behaviour of the free agents. For example, many real-world scenarios could be better described with a stochastic search process. One might also consider search strategies taken from cognitive psychology, like the famous *satisficing* heuristic of Simon (1957) or the *take-the-best* heuristic of Gigerenzer and Goldstein (1996). It may be a worthwhile effort to perform a similar analysis like the one presented in this paper, but with alternative behavioural assumptions for the free agents. Models which combine rational and boundedly rational agents are very rare in the game theoretic literature,[22] yet in reality an increasing number of situations of this kind can be found. For example, in stock exchange markets, humans trade simultaneously with computer programs. The computers act extremely fast, without any psychological biases, and they have

[22]Indeed, we do not know of any papers on the subject.

superior computing power – hence they could be considered to be fully rational players.

Despite of its various reasonable alternatives, we want to stress that the search process modelled in this article has some intriguing features. Firstly, it is quite natural to assume that the free agents check for free tasks in order of their preferences. Secondly, as we mentioned in Section 3, a result of McVitie and Wilson (1971) implies that the order in which the free players approach tasks and get deprived of tasks by other players does not influence the outcome of the process. This gives our search process considerable robustness with regard to many specific circumstances prevailing in real-life scenarios. Thirdly, in the deferred acceptance algorithm of Gale and Shapley (1962) there is no incentive for the proposing side, in our case the free agents, to misrepresent their preferences (cf. Dubins and Freedman (1981), Roth (1982)). In our context, this means that the free agents cannot improve their outcome by changing the order in which they approach tasks. So even if free agents would have enough information and computing power to act strategically, it would not be worthwhile to do this. In contrast, alternative models of search behaviour would have to take care of strategic manipulations on the free agents' parts. Of course, this makes handling our model merely *convenient*, yet does not say anything about the validity of its assumptions.

Other modifications to our model come to mind. It may be interesting to change the informational assumptions of the model. What if the productivities of the autonomous workers for different tasks is private knowledge of that worker?[23] Would there be a way to make the free agents reveal their private information? Could they even be incentivised to pick the task which would be best from the central planner's point of view? Designing a transfer scheme to achieve such goals would demand the free agents to be modelled with cardinal preferences. This would reduce the robustness of the model but it might add economically interesting dynamics similar to those which can be found in the famous labour market adjustment models of Crawford and Knoer (1981) and Kelso and Crawford (1982).

The idea of introducing autonomous agents in scenarios where the central planner normally has full control is by no means limited to the allocation domain considered here. In fact, many other standard problems can be extended to include autonomous agents. Transportation or network flow with some transfers performed by autonomous agents, knapsack where autonomous agents are able to add their own items to the knapsack, and graph colouring with some nodes coloured by the agents are just a few examples.

References

Y. Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* Yale University Press, 2006.

[23]We thank Ulrich Pferschy for suggesting this modification.

D. C. Brabham. Crowdsourcing as a Model for Problem Solving. *Convergence*, 14(1):75–90, 2008.

J. R. Brown. The Knapsack Sharing Problem. *Operations Research*, 27(2): 341–355, 1979.

B. Colson, P. Marcotte, and G. Savard. An Overview of Bilevel Optimization. *Annals of Operations Research*, 153:235–256, 2007.

V. P. Crawford and E. M. Knoer. Job Matching with Heterogeneous Firms and Workers. *Econometrica*, 49:437–450, 1981.

L. E. Dubins and D. A. Freedman. Machiavelli and the Gale-Shapley Algorithm. *American Mathematical Monthly*, 88:485–494, 1981.

D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, 69:9–15, 1962.

G. Gigerenzer and D. G. Goldstein. Reasoning the Fast and Frugal Way: Models of Bounded Rationality. *Psychological Review*, 103:650–669, 1996.

G. Gigerenzer and P. M. Todd. *Simple Heuristics that make us smart.* Oxford University Press, 2000.

J. Howe. *Crowdsourcing: Why the Power of the Crowd is driving the Future of Business.* Crown Business, 2008.

A. S. Kelso and V. P. Crawford. Job Matching, Coalition Formation, and Gross Substitutes. *Econometrica*, 50(6):1483–1504, 1982.

D. E. Knuth. *Stable Marriage and its Relation to other Combinatorial Problems.* American Mathematical Society, 1997.

T. C. Koopmans and M. Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, 25(1):53–76, 1957.

H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

S. Martello. Jenő Egerváry: From the Origins of the Hungarian Algorithm to Satellite Communication. *Central European Journal of Operations Research*, 18:47–58, 2010.

D. G. McVitie and L. B. Wilson. The Stable Marriage Problem. *Communications of the ACM*, 14:486–492, 1971.

J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

J. von Neumann. A certain Zero-sum Two-person Game equivalent to the Optimal Assignment Problem. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, pages 5–12. Princeton University Press, 1953.

J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

I. A. Nikolic and H. Maikisch. Public-Private Partnerships and Collaboration in the Health Sector. *HNP Discussion Paper*, 2006. (`http://info.worldbank.org/etools/docs/library/240103/PUBLIC~2.PDF`).

M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

U. Pferschy and J. Schauer. The Knapsack Problem with Conflict Graphs. *Journal of Graph Algorithms and Applications*, 13:233–249, 2009.

J. A. Riis. *Hero Tales of the Far North.* Echo Library, 2007.

A. E. Roth. The Economics of Matching: Stability and Incentives. *Mathematics of Operations Research*, 7:617–628, 1982.

A. E. Roth and M. Sotomayor. *Two-Sided Matching - A Study in Game Theoretic Modeling and Analysis.* Cambridge University Press, 1990.

U. G. Rothblum. Characterization of Stable Matchings as Extreme Points of a Polytope. *Mathematical Programming*, 54:57–67, 1992.

H. A. Simon. *Models of Man: Social and Rational.* Wiley, 1957.

S.-K. Templeton. Dentists refuse to treat bad Teeth. *The Sunday Times*, May, 27, 2007.

J. H. Vande Vate. Linear Programming brings Marital Bliss. *Operations Research Letters*, 8(3):147–153, 1989.

T. Yamada and M. Futakawa. Heuristic and Reduction Algorithms for the Knapsack Sharing Problem. *Computers & Operations Research*, 24(10):961–967, 1997.

T. Yamada, S. Kataoka, and K. Watanabe. Heuristic and Exact Algorithms for the Disjunctively constrained Knapsack Problem. *Information Processing Society of Japan Journal*, 43:2864–2870, 2002.

G. Yu. On the Max-Min 0-1 Knapsack Problem with Robust Optimization Applications. *Operations Research*, 44(2):407–415, 1996.

Appendix A: Formal statement of the DAB algorithm

Here we will formally state the search process of the free agents which was informally described in Section 3.

The search process of the free players coincides with the stable matching algorithm of Gale and Shapley (1962) if the free agents stand for the men and the tasks stand for the women. However, the algorithm is executed in a marriage market in which all those tasks which were assigned to coordinated agents are not available anymore. This idea is formalised below.

A coordinated assignment is a matching $\mu_C \subseteq C \times T$ (no free player f is a member of any pair). We denote by $(F, T, \succ_{\boldsymbol{F}})_{\mu_C}$ a marriage market[24] formed by free agents and tasks in which those tasks which were matched under μ_C are not available anymore. Formally:

$$(F, T, \succ_{\boldsymbol{F}})_{\mu_C} = (F, T \setminus \{t \mid (c,t) \in \mu_C\}, \succ).$$

The following algorithm is the deferred acceptance algorithm of Gale and Shapley (1962) executed in the market $(A, T, \succ_{\boldsymbol{F}})_{\mu_C}$. The output matching μ^{SAAP} of the algorithm is a feasible assignment μ^{SAAP} for the underlying SAAP. The matching $\mu_F[\mu_C] := \mu^{\text{SAAP}} \setminus \mu_C$ is the free agent optimal stable matching in the market $(A, T, \succ_{\boldsymbol{F}})_{\mu_C}$.[25]

[24]Marriage markets are formally defined on page 6.

[25]For details cf. Gale and Shapley (1962) and Roth and Sotomayor (1990), Theorem 2.12., p. 32.

Deferred acceptance algorithm with blocked tasks (DAB)

Input:
A marriage market $(A, T, \succ_{\boldsymbol{F}})_{\mu_C}$.

Initialisation:
Set $i := 0$. For each $f \in F$, let there be a set $C_0(f) = \emptyset$. Set $\mu_0 := \mu_C$, $i := i + 1$ and go to the main iteration.

Main iteration:
Define a set

$$\mu_i^a := \mu_{i-1} \cup \{(f,t) \in F \times T \mid t = \max_{\succ_f} \mathcal{T}_f \setminus C_{i-1}(f)\}$$

Check: If μ_i^a is an assignment, stop the algorithm and set $\mu^{\text{SAAP}} := \mu_i^a$ and $\mu_F[\mu_C] := \mu^{\text{SAAP}} \setminus \mu_C$. Otherwise, define

$$\begin{aligned} C_i(f) \quad := & \\ & C_{i-1}(f) \cup \\ & \{t \in \mathcal{T}_f \mid (f,t) \in \mu_i^a : \ \exists (\hat{f}, t) \in \mu_i^a \text{ with } v((\hat{f},t)) > v((f,t)) \text{ for } \hat{f} \in F\}. \end{aligned}$$

Remove (in an arbitrary order) all (f,t) from μ_i^a with $t \in C_i(f)$ and denote the resulting set by μ_i. Set $i := i + 1$ and return to the start of the main iteration.

Lemma 1. *At any stage j of the DAB-algorithm, $t \in C_j(f)$ for $f \in F$ implies $(f,t) \notin \mu_j$.*

Proof. If $(f,t) \in \mu_j^a$, then (f,t) gets removed from μ_j^a at that same stage j and the resulting matching is μ_j. However, if $(f,t) \notin \mu_j^a$ in the first place, then of course $(f,t) \notin \mu_j$. □

Lemma 2. *Each execution of the DAB-algorithm stops after a finite number of stages.*

Proof. At any stage $i \geq 1$ of the algorithm we have $C_{i-1}(f) \subseteq C_i(f)$ for all $f \in F$. We will show that at a stage i at which μ_i^a is no assignment, there is at least one player f with $C_{i-1}(f) \subset C_i(f)$ (strict inclusion). This concludes the proof by a potential function argument. If at stage i the set μ_i^a is not an assignment (and thus the algorithm has not stopped at the beginning of stage i), there are two distinct pairs $(f,t), (\hat{f},t) \in \mu_i^a \cap F \times T$ which share the position t. Condition 3.1 (page 6) ensures $v((f,t)) \neq v((\hat{f},t))$. W.l.o.g. assume $v((f,t)) > v((\hat{f},t))$. Then by definition of $C_i(\hat{f})$ holds $t \in C_i(\hat{f})$. We finish by showing $t \notin C_{i-1}(\hat{f})$. By contradiction, assume $t \in C_{i-1}(\hat{f})$. Then by Lemma 1, $(\hat{f},t) \notin \mu_{i-1}$. As $(\hat{f},t)$ in μ_i^a, it must hold $t = \max_{\succ_{\hat{f}}} \mathcal{T}_{\hat{f}} \setminus C_{i-1}(\hat{f})$. As we assumed $t \in C_{i-1}(\hat{f})$, this is impossible. □

Lemma 3. *For a given coordinated assignment μ_C, an execution of the DAB-algorithm yields a unique output assignment μ^{SAAP}.*

Proof. At an arbitrary stage i of the algorithm, only the order of removal of those pairs (f,t) with $t \in C_i(f)$ from the set μ_i^a is not uniquely determined. However, here it is clear that for a given position t, *all* pairs (f,t) will be removed for which $(f,t) \neq \max_{(\hat{f},t) \in \mu_i^a \cap F \times T} v((\hat{f},t))$. So the order of removal does not have any impact on the set μ_i, which is constructed on that stage. $\square$

Appendix B: Decomposition of the program (5.5) into first-level and second-level optimisation problems

In this appendix we will explicitly state the first-level maximisation and the second-level minimisation of (5.5) as two distinct programs. We denote those variables $x(m)$ for which $m \in C \times T$ by $x_c(m)$, variables $x(m)$ for which $m \in F \times T$ by $x_f(m)$, and variables $x(m)$ for which $m \in A \cup T$ by $x_u(m)$. In the first level, CP sets $x_c(\cdot)$ to assign coordinated agents to tasks keeping in mind that in the second level the free agents will follow DAB to allocate remaining tasks among themselves. The latter generates a configuration of the variables $x_f(\cdot)$. Let $g(x_c)$ denote the value $v(\mu_F[\mu_C])$, where μ_C is derived from the configuration of the variables $x_c(\cdot)$, and $\mu_F[\mu_C]$ is derived from the configuration of the variables $x_f(\cdot)$.

For each task (coordinated agent) we define the set of coordinated agents (tasks) that it can be matched with:

$$\mathcal{M}_t^c := \{(a,\hat{t}) \in C \times T \mid \hat{t} = t\}$$
$$\mathcal{N}_a^c := \{(\hat{a},t) \in C \times T \mid \hat{a} = a\}.$$

We do the same for free agents including the possibility for the task (agent) to remain unmatched:

$$\mathcal{M}_t^f := \{t\} \cup \{(a,\hat{t}) \in F \times T \mid \hat{t} = t\}$$
$$\mathcal{N}_a^f := \{a\} \cup \{(\hat{a},t) \in F \times T \mid \hat{a} = a\}.$$

First-level maximisation

$$\max_{x_c(m): m \in C \times T} g(x_c) + \sum_{m \in C \times T} x_c(m) v(m)$$

$$\forall m \in C \times T \qquad x_c(m) \in \{0,1\}$$

$$\forall t \in T: \qquad \sum_{m \in \mathcal{M}_t^c} x_c(m) \leq 1 \tag{B.1}$$

$$\forall a \in C: \qquad \sum_{m \in \mathcal{N}_a^c} x_c(m) \leq 1 \tag{B.2}$$

The objective function consists of two terms: the value of the coordinated assignment and the value $g(x_c)$ of the tasks performed by the free agents. The constraints ensure the matching is valid. As noted before, DAB leads to the least preferred stable matching for the tasks. So $g(x_c)$ can be computed by the following integer program which chooses a stable matching with the lowest value.

Second-level minimisation

$$\begin{aligned}
g(x_c) = \quad & \min_{x_f(m):m\in F\times T} \sum_{m\in F\times T} x_f(m)v(m) \\
\forall m \in (F\times T)\cup F\cup T: \quad & x_f(m) \in \{0,1\} \\
\forall t \in T: \quad & \sum_{m\in\mathcal{M}_t^f} x_f(m) = 1 - \sum_{m\in\mathcal{M}_t^c} x_c(m) && \text{(B.3)} \\
\forall a \in F: \quad & \sum_{m\in\mathcal{N}_a^f} x_f(m) = 1 && \text{(B.4)} \\
\forall (m', m'') \in E: \quad & x_f(m') + x_f(m'') \le 1 && \text{(B.5)}
\end{aligned}$$

A reader may notice that the constraints ensuring the matching is valid are weak in the first level (Equations B.1 and B.2) but strict in the second (Equations B.3 and B.4). A weak constraint in the first-level maximisation allows a task (or an agent) to be unassigned, while a strict would not. In fact, free agents and tasks unassigned after the first stage can remain unassigned in the second-level problem, in which case the variable $x_f(m) \mid m \in F \cup T$ is set to one. Expanding the set of second-level variables to include $x_f(m) \mid m \in F \cup T$ enables a concise representation of the conflict set constraint B.5.