Incentivizing Cooperation between Heterogeneous Agents in Dynamic Task Allocation

Sofia Amador Nelke
Department of Industrial Engineering and Management,
Ben-Gurion University of the Negev,
Beer-Sheva, Israel
amador@post.bgu.ac.il

Roie Zivan
Department of Industrial Engineering and Management,
Ben-Gurion University of the Negev,
Beer-Sheva, Israel
zivanr@post.bgu.ac.il

ABSTRACT

Market Clearing is an economic concept that features attractive properties when used for resource and task allocation, e.g., Pareto optimality and Envy Freeness. Recently, an algorithm based on Market Clearing, FMC_TA, has been shown to be most effective for realistic dynamic multi agent task allocation, outperforming general optimization methods, e.g., Simulated annealing, and dedicated algorithms, specifically designed for task allocation. That been said, FMC_TA was applied to a homogeneous team of agents and used linear personal utility functions for representing agents' preferences. These properties limited the settings on which the algorithm could be applied.

In this paper we advance the research on task allocation methods based on market clearing by enhancing the FMC_TA algorithm such that it: 1) can use concave personal utility functions as its input and 2) can apply to applications which require the collaboration of heterogeneous agents, i.e. agents with different capabilities. We demonstrate that the use of concave functions indeed encourages collaboration among agents. Our results on both homogeneous and heterogeneous scenarios indicate that the use of personal utility functions with small concavity is enough to achieve the desired incentivized cooperation result, and on the other hand, in contrast to functions with increased concavity, does not cause a severe delay in the execution of tasks.

Keywords

Cooperation and coordination, Task Allocation, Fisher Market

1. INTRODUCTION

Realistic multi-agent dynamic task allocation often include tasks that require cooperation between agents. In some cases this may be because the task is too complex to be handled by a single agent. In other cases the task may require multiple skills, which no single agent possesses [6]. Furthermore, multiple agents collaborating in performing a task can significantly reduce the time required to complete it, even if the task does not strictly require such cooperation.

The law enforcement problem is such a realistic application, where police officers must attend to events reported to the police headquarters. While some of these events are simple and can be handled by a single police unit (e.g., domestic quarrel, small car accident), more complex events (e.g., bank robbery, terror attack) require multiple officers to work together. Such complex events many times require cooperation between officers with different skills, e.g., breaking into a building with hostages, defusing bombs etc.

In [1], a task allocation algorithm based on Fisher Market Clearing (FMC_TA) was proposed. The algorithm is suitable for dynamic task allocation with spatial and temporal constraints in both centralized and distributed settings, and requires worst-case polynomial and pseudo-polynomial time respectively. The algorithm was applied to a law enforcement problem, and the empirical results reported demonstrate its advantage in comparison with state of the art algorithms, e.g., Market-based task allocation [5], Coalition Formation with Look-Ahead [7], Simulated Annealing [8] and general-purpose algorithms solving a distributed constraint optimization problem(DCOP) [11].

The Fisher Market [4] receives as input a matrix of numeral preferences, i.e., the linear utility the agent derives when allocated the resource/task as a function of the portion of the resource/task it receives[1]. The output of a market clearing algorithm is an allocation of the tasks to agents that specifies the fraction of each task, which is allocated to each of the agents [2]. FMC_TA uses this output to allocate the tasks to agents and then generates the order in which the agents will perform the tasks. This order must include coordinated arrival to tasks that are shared and need to be performed concurrently by multiple agents.

While the success of the FMC_TA algorithm in comparison to both general optimization algorithms and specifically designed task allocation algorithms was encouraging, two of its properties limited the scenarios to which it could be applied.

First, the utility functions that were used as the input to the Fisher market in FMC_TA were linear [1], i.e., the mechanism considers the utility that an agent derives when performing some task to be proportional to the portion of the task allocated to it. The main problem with such linear functions is the difficulty to express cooperation requirements for performing a task. Thus, in applications in which mutual performance of tasks is essential and should be preferred over allocations of tasks to single agents, the mechanism does not allow to express these preferences.

Second, the mechanism was designed for homogeneous agents, i.e., all agents were assumed to have the same skills. Tasks that require agents with special skills (e.g., officers that can defuse bombs) could not be represented and consequently, the algorithm could not guarantee that the agents assigned to such tasks, have the capabilities required for performing them.

In this paper we address the two limitations of FMC_TA mentioned above and design FMC_TA^{+}, which can use concave per-

[1]Since in this paper we are focusing on realistic task allocation applications, we will only consider tasks as the products of the Fisher Market

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

sonal utility functions as input. We hypothesize that by replacing linear functions by concave functions, we will be able to incentivize agents to cooperate [3]. Furthermore, we design FMC_TA^{H+} such that it allows to represent agent's skills and express the combination of skills required to perform a task.

In more details, we make four major contributions:

1. We propose the use of concave utility functions within FMC_TA (FMC_TA^{+}), including the details of the implementation and a demonstration of how the use of concave personal utility functions encourages collaboration among agents, in comparison with using standard linear utility functions, as input for the market clearing mechanism.

2. For a scenario including two police units and two tasks, we analyze the advantages and disadvantages in cooperation and specify the conditions for cooperation to be more efficient than individual work, i.e., we indicate the problem settings that lead to higher social welfare while sharing a task and analyze the implication on execution delay of the corresponding missions. These insights are used to design a heuristic for the general case that allows our algorithm to determine for which tasks it should incentivize cooperation using concave personal utility functions, as input for the FMC mechanism.

3. We extend the expressiveness LEP model presented in [1] by proposing LEP^{H}, which allows the representation of tasks that handling them requires specific combinations of agent's skills, and heterogeneous agents, each with a possibly different set of skills.

4. We design FMC_TA^{H+}, an algorithm that can allocate agents to tasks such that their skills will apply to the combination of skills required to perform the tasks. We overcome the limitation in FMC_TA by treating each skill required for a task as an independent product in the Fisher market that is attractive only for agents that have this skill. Thus, the outcome of the FMC allocation must include an agent with such a skill to participate in performing the corresponding task.

We present empirical evaluation that emphasize the significance of the extensions we propose to FMC_TA. Our results when comparing FMC_TA^{+} to FMC_TA indicate that small concavity is enough to encourage the agents to cooperate and achieve the desired outcome in the level of cooperation and team utility. In addition, when using functions with small concavity the execution delay is not increased as much as when using functions with high concavity.

In order to evaluate the performance of FMC_TA^{H+} on heterogeneous dynamic task allocation problems, we designed an experiment that applies to the extended LEP^{H} and compared FMC_TA^{H+} with Simulated annealing (SA), a state of the art meta-heuristic, which performed best among all other competing heuristics, both in our experiments and in the experiments on the homogeneous settings presented in [1]. The results present a significant advantage of the performance of FMC_TA^{H+} compared to the performance of SA in multiple parameters.

2. BACKGROUND

This section describes the main details of the LEP formalization and the FMC_TA algorithm. For further details see [1].

2.1 Law Enforcement Problems (LEP)

A static LEP includes n cooperative agents (police units), $a_1, \ldots, a_n \in A$ and m tasks $v_1, \ldots, v_m \in V$ situated in a city, with the set of all possible locations denoted by L. We use $\rho(a_i, v_j)$ to denote the travel time from agent a_i's current location to task the location of task v_j.

Each task v_j has an *arrival time* $\alpha(v_j)$, an *importance* $I(v_j) > 0$ and a *workload* $w(v_j)$ specifying how much work (in time units) must be performed to complete the task. Multiple agents can share a single event.

An allocation of tasks to agents is denoted by an $n \times m$ matrix X where entry x_{ij} is the fraction of task v_j that is assigned to a_i. The tasks allocated to agent a_i must be scheduled , i.e., $\sigma^i = (v^i_1, t^i_1, t^{i\prime}_1), \ldots, (v^i_{M_i}, t^i_{M_i}, t^{i\prime}_{M_i})$ is a sequence of M_i triples of the form $(v^i_k, t^i_k, t^{i\prime}_k)$, where v^i_k is the task performed by a_i, starting at time t^i_k until time $t^{i\prime}_k$.

The utility for performing task v_j depends on the number of agents q that work simultaneously on task v_j and is denoted by the non-negative *capability* function, $Cap(v_j, q)$. Let $d_q^{v_j}$ be the time that q agents are working together on mission v_j. Thus, $\frac{q d_q^{v_j}}{w(v_j)}$ is the relative part of the mission that is performed by q agents (as mentioned above, $w(v_j)$ is the total time required to complete the mission). The utility derived for completing task v_j is:
$\sum_{q=1}^{n} \frac{q d_q^{v_j}}{w(v_j)} Cap(v_j, q)$.

In addition, the *soft deadline* function $\delta(t) = \beta^{\gamma t}$, where $\beta \in (0, 1]$ and $\gamma \geq 0$ are constants, reduces the utility as a result of a delay in the starting time of handling a task.

The discounted utility for performing task v_j that arrives at time $\alpha(v_j)$ and initially is handled at time t_{v_j} is:
$U'(v_j) = \beta^{\gamma(t_{v_j} - \alpha(v_j))} \sum_{q=1}^{n} \frac{q d_q^{v_j}}{w(v_j)} Cap(v_j, q)$

When a new task arrives, the *current task* (if any) being performed by agent a_i is denoted CT_i. Agents can *interrupt* the performance of their current task. Task interruption incurs a *penalty*, $\pi(v_j, \Delta w)$, which depends on the task v_j and the amount of work Δw completed when the task is interrupted. The penalty for event v_j decreases exponentially with Δw to a minimum value:
$\pi(v_j, \Delta w) = \max\{I(v_j) c^{w(v_j) - \Delta w}, \phi \cdot I(v_j)\}$, where $c \in [0, 1)$ and $\phi > 0$ are constants and $\phi I(v_j)$ is the minimum penalty.

The total utility derived for performing v_j is thus

$$U(v_j) = U'(v_j) - \sum_{a_i : v^i_1 \neq CT_i} \pi(CT_i, \Delta w)$$

2.2 FMC-based Task Allocation

A Fisher market [2] contains n buyers, each endowed with an amount of money, and m goods. An $n \times m$ matrix R represents the preferences of buyers over products. A market-clearing solution is a price vector p specifying a price p_j for each good j that allows each buyer i to spend all her money on goods that maximize bang-per-buck (r_{ij}/p_j) while all goods are sold. An FMC allocation is an $n \times m$ matrix X where each entry $0 \leq x_{ij} \leq 1$ is the fraction of good j allocated to buyer i given the market-clearing prices p. FMC allocations are Pareto optimal and also envy-free when monetary endowments are equal [9].

FMC_TA (Fisher Market Clearing-based Task Allocation) represents agents and tasks as buyers and goods, respectively, and endows each agent with an equal amount of money. R is constructed by optimistically ignoring the inter-task ordering constraints and assuming the maximum value for the capability function. Specifically, we set entry r_{ij} at time t to be the utility of a_i immediately moving to v_j and performing it with the optimal number of agents:

$$r_{ij} = \beta^{\gamma(t + \rho(a_i, v_j))} \max_q \{Cap(v_j, q)\} - \pi(CT_i, \Delta w)\}$$

where the penalty is omitted if $CT_i = v_j$.

In the second stage of FMC_TA, the allocated tasks are scheduled for each agent to reflect the spatio-temporal inter-task and inter-agent constraints. The tasks allocated to each agent a_i are ordered by greedily prioritizing according to $r_{ij}/x_{ij}w_j$, i.e., the ratio between the utility the agent derives for performing a task (as reported to the mechanism) and the time the agent spends on the task. Once the initial order is selected, each agent computes the initial schedule σ_i by setting t^i_k and $t^{i\prime}_k$. Then, start times are updated to reflect inter-agent constraints so that shared task execution leads to higher social welfare and consequently to better performance. Next, a check is performed whether the order of tasks in the individual schedules can be optimized by moving individual tasks earlier without delaying the execution of shared tasks.

This process, including the FMC allocation and the scheduling of tasks, is completed in polynomial time at the worst case [1].

3. CONCAVE PERSONAL UTILITY FUNCTIONS

The utility functions used as input for the FMC_TA algorithm, as described above, are limited to expressing unary preferences of agents over products in the Fisher market. Amador et. al. demonstrate that, although the input to the Fisher market does not allow to represent spatial and temporal constraints, the fairness and Pareto optimality properties of the allocation overcome this limitation and produce high quality results [1]. However, when using linear functions, benefits that agents can derive by cooperating cannot be expressed, i.e., the input matrix is limited to indicating the maximal utility an agent can derive by completing a task, but it does not allow to indicate that higher utility is derived if the task is shared by a number of agents.

In more details, in the R matrix that is used as input for the FMC_TA algorithm, each entry r_{ij} contains a numeric number that is used as a linear utility function, i.e., the mechanism considers the utility for agent a_i from allocation x_{ij} to be $r_{ij}x_{ij}$ (recall that x_{ij} is the portion of task j allocated to agent i). The FMC mechanism allows agents only to be assigned tasks that maximize bang-per-buck according to the price vector, thus, an agent is indifferent regarding the portions of tasks it receives. In other words, the agent values equally allocations in which it shares tasks with others and allocations in which it performs tasks on its own (as long as all tasks are among the set of tasks that maximize bang per buck).

In order to overcome the limitation specified above, we propose the use of concave utility functions as input to the FMC_TA algorithm. Intuitively, if we consider equal size portions of a task, concave utility functions offer agents more utility for the first portion of the task they are allocated, than for the next portion. The utility becomes smaller (i.e., diminishes) with every additional portion they allocated to them [3].

For an agent a_i and a task v_j, the concave utility function $u_{ij}(x_{ij}) : [0,1] \rightarrow [0, r_{ij}]$ is monotonically increasing in x_{ij}. The parameter r_{ij} is the original entry from the R matrix, i.e., for agent a_i, the agent's utility for allocation $x_{ij} = 1$ (entire mission) is equal regardless of the concavity of the function ($u_{ij}(1) = r_{ij}$). The concave utility function is of the form: $u_{ij}(x_{ij}) = (h_{ij}x_{ij})^{\mu_j}$ where $\mu_j \in (0,1]$ and $h_{ij} = (r_{ij})^{1/\mu_j}$.

The following example demonstrates how the use of a concave utility function incentivizes agents to share tasks. Figure 1 includes three curves, each representing the utility an agent derives as a function of the portion x of a task which is allocated to it, i.e., $x \in [0,1]$. All three functions are concave utility functions with distinct exponent value $\mu = 0.25, 0.5, 1$. For sake of convenience, and with no contradiction to the definition of the concave function stated above, we denote $u(x) = (hx)^{\mu} = u_{\mu}(x)$. Obviously, the linear utility function is a special case of the concave utility function where $\mu = 1$. For the same value of h the exponent μ affects the gradient. The gradient is steeper in the first part of the function, as μ is closer to zero. The utility for a fraction x is greater when μ is closer to zero. Formally, $u_{\mu'}(x) > u_{\mu}(x)$ when $\mu' < \mu$. Thus, for example when $x = 0.4$ the utilities are $u_1(0.4) < u_{0.5}(0.4) < u_{0.25}(0.4)$.

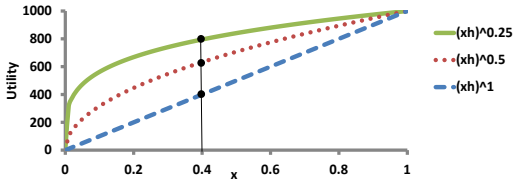

Figure 1: Concave utility function vs. Linear utility function.

Notice, that the functions described above only consider the portion of a task that a single agent is allocated and ignore the portions allocated to other agents. Hence, when presented a choice between performing a fraction of a single task or performing the same fraction split between two tasks, assuming all tasks have the same utility function, the structure of the function encourages an agent to prefer performing smaller portions of two tasks. Formally, $u_{\mu'}(x) < u_{\mu}(x_1) + u_{\mu}(x_2)$ when $x_1 + x_2 = x$. Thus, when using concave functions (with $\mu < 1$) the mechanism tends to split tasks between agents, even in scenarios where there are many tasks (more than the number of agents).

Two parameters of the function, h and μ, enable representing the importance of the task (that takes into account spatial and temporal constraints) and the specific importance of cooperation, respectively. For tasks where cooperation is not required we use $\mu_j = 1$ and as the need for cooperation increases the μ parameter decreases.

3.1 Benefit Bounds in a Two over Two LEP

In this section we analyze the benefit agents derive from cooperation in a small LEP including two police units and two missions (tasks).[2] We indicate the condition under which cooperation is more rewarding than individual work. While the utility agents derive from a task often increases when the task is shared, the execution delay may also grow as a result of sharing. Thus, we also analyze and detect the lower bound for the differences in execution delay between cooperative and non-cooperative execution.

The scenario we analyze includes two missions (tasks), v_1, v_2, and two agents, a_1, a_2. Both missions arrive at time $t = 0$ and for both, cooperative execution is more rewarding. The reward for execution of task v_j is generally defined for each mission j as:

$$Cap(v_j, q) = \begin{cases} u_j & \text{if } q \geq 2 \\ u_j c_j & \text{if } q = 1 \end{cases}$$

where $c_j < 1$. For each task v_j there is a workload w_j. We use the $\rho(a, v)$ function to denote distance in time units between two locations. For the case of individual work, we assume that mission v_i is closer to agent a_i, and therefore, it attends that mission i.e., a_1 attends v_1 and a_2 attends v_2. For the cooperation alternative execution, without loss of generality, we assume that agents first handle mission v_1. We choose to relate to the case where cooperation is possible in terms of time, i.e., the second agent can arrive at the mission location before the first agent finishes handling it.

[2]For larger scenarios we present empirical results in the Section 5, indicating the effect of increased cooperation.

Formally:

$$\rho(a_1, v_1) + w_1 > \rho(a_2, v_1) \quad (1)$$

First we present the average execution delay for each case. We denote the average execution delay for individual and cooperative work by ED_{ind} and ED_{coo}, respectively. For the case of individual work the execution delay for each mission is equal to the travel time from the initial location of the agent that is assigned this mission to the location of the mission $\rho(a_i, v_i)$. This scenario is illustrated in figure 2. Thus, in a scenario without cooperation the average execution delay is:

$$ED_{ind} = \frac{\rho(a_1, v_1) + \rho(a_2, v_2)}{2} \quad (2)$$

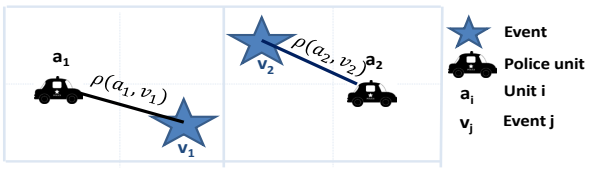

Figure 2: Individual work example.

For the case in which agents cooperate in performing tasks we assume, without loss of generality, that agent a_1 is closer to mission v_1 and will arrive first at the location. The execution delay for mission v_1 remains $\rho(a_1, v_1)$. Before executing mission v_2 the agents have to complete mission v_1. The execution time is composed of individual work time of agent a_1 and cooperative work, after agent a_2 arrives. The individual work time is the difference between the arrival times of the agents to the mission, $\rho(a_2, v_1) - \rho(a_1, v_1)$, and thus, the simultaneous work time is: $\frac{w_1 - (\rho(a_2,v_1) - \rho(a_1,v_1))}{2}$. Next, the agents travel to the location of mission v_2. It takes them $\rho(v_1, v_2)$ time units. This scenario is illustrated in figure 3 and the corresponding average execution delay is:

$$ED_{coo} = \frac{2\rho(a_1, v_1) + (\rho(a_2, v_1) - \rho(a_1, v_1))}{2} + \frac{\frac{w_1 - (\rho(a_2,v_1) - \rho(a_1,v_1))}{2} + \rho(v_1, v_2)}{2} \quad (3)$$

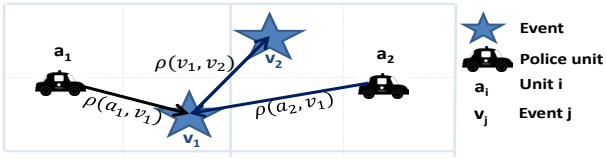

Figure 3: Cooperative work example.

PROPOSITION 1. $ED_{coo} - ED_{ind} \geq 2\rho(v_1, v_2)$

PROOF. Following equations 2 and 3 we get the following:

$$\rho(a_2, v_1) + \rho(a_1, v_1) + w_1 + 2\rho(v_1, v_2) > 2\rho(a_2, v_2) \quad (4)$$

According to condition 1, $\rho(a_1, v_1) + w_1 > \rho(a_2, v_1)$. According to the assumption that mission v_2 is closer to agent a_2, $\rho(a_2, v_1) > \rho(a_2, v_2)$. Thus equation 4 is valid and the lower bound for the difference is $2\rho(v_1, v_2)$. □

Next we analyze the difference in social welfare between the two executions. In order to express the utility for each mission we have to specify the soft deadlines (as defined in [1]), i.e., the reduction in social welfare as a result of the execution delay discussed above. We denote soft deadline functions for events by $\delta(t) = \beta^t$ where $\beta \in (0, 1]$ and t is the execution delay. We denote the social welfare for individual and cooperative work SW_{ind} and SW_{coo}, respectively. For the case of individual work the reward for each mission is $u_i c_i$. Thus, the social welfare in the case of individual work is:

$$SW_{ind} = \beta^{\rho(a_1,v_1)} u_1 c_1 + \beta^{\rho(a_2,v_2)} u_2 c_2 \quad (5)$$

In the cooperative case, the agents cooperate on both missions so they earn the maximum reward, u_i. However, for mission v_1 there is a period of time that agent a_1 handles the mission alone until agent a_2 arrives. The length of this period is $(\rho(a_2, v_1) - \rho(a_1, v_1))$, and it is denoted by t_{a_1}. The reward for that period is $u_1 c_1$. Thus, the social welfare for cooperative execution is:

$$SW_{coo} = \beta^{\rho(a_1,v_1)}(u_1 c_1 \frac{t_{a_1}}{w_1} + u_1(1 - \frac{t_{a_1}}{w_1})) + \beta^{\rho(a_1,v_1) + \frac{w}{2} + \frac{t_{a_1}}{2} + \rho(v_1,v_2)} u_2 \quad (6)$$

PROPOSITION 2. *The difference, in social welfare between the cooperative mode and individual mode is:*

$$SW_{coo} - SW_{ind} = diff_{sw} = u_1 \beta^{\rho(a_1,v_1)}(1 - \frac{t_{a_1}}{w_1})(1 - c_1) - u_2(c_2 \beta^{\rho(a_2,v_2)} - \beta^{\rho(a_1,v_1) + \frac{w}{2} + \frac{t_{a_1}}{2} + \rho(v_1,v_2)}) \quad (7)$$

PROOF. Follows immediately from equations 5 and 6. □

After establishing the social welfare derived for both executions, we need to determine when cooperative execution is beneficial, i.e., when $diff_{sw} > 0$. To this end we analyze $diff_{sw}$ as a function of two parameters and establish two corollaries that follow immediately from Proposition 2.

The first is the ratio between the maximum utility that can be derived from the two missions, $\frac{u_1}{u_2}$. Note that $diff_{sw}$ increases as the ratio $\frac{u_1}{u_2}$ increases.

COROLLARY 1. $diff_{sw} > 0$ *when*

$$\frac{u_1}{u_2} > \frac{c_2 \beta^{\rho(a_2,v_2)} - \beta^{\rho(a_1,v_1) + \frac{w}{2} + \frac{t_{a_1}}{2} + \rho(v_1,v_2)}}{\beta^{\rho(a_1,v_1)}(1 - \frac{t_{a_1}}{w_1})(1 - c_1)} \quad (8)$$

The second parameter that we investigate is the distance between the missions $\rho(v_1, v_2)$. $diff_{sw}$ decreases as the distance $\rho(v_1, v_2)$ increases, i.e., the farther away the missions are from each other the better it is for agents to separate. The condition on $\rho(v_1, v_2)$ that causes this inequality $diff_{sw} > 0$ to be true is:

COROLLARY 2. $diff_{sw} > 0$ *when*

$$\rho(v_1, v_2) < \log_\beta \frac{u_1 \beta^{\rho(a_1,v_1)}(1 - \frac{t_{a_1}}{w_1})(1 - c_1) - u_2 c_2 \beta^{\rho(a_2,v_2)}}{u_2 \beta^{\rho(a_1,v_1) + \frac{w}{2} + \frac{t_{a_1}}{2}}} \quad (9)$$

Although the complexity of the problem prevents us from producing such analytical results for the general case, the properties proved above can be used to design a heuristic for determining for which tasks cooperation should be incentivized by using concave personal utility functions as input for the FMC mechanism. In detail, let dt be *distance threshold* and rt be a *ratio threshold*. For every task v_j that requires cooperation we check the following two conditions:

1. $\min_{v_k}(\rho(v_j, v_k)) < dt$
2. $\max_{v_k}(\frac{u_{v_j}}{u_{v_k}}) < rt$

If at least one of the conditions above holds, we encourage cooperation by using a concave utility function for this task in FMC, otherwise we use a linear utility function.

4. HETEROGENEOUS AGENTS

In many realistic dynamic task allocation scenarios the agents are heterogeneous, i.e., each agent has a set of skills, which can be different from the sets of skills of other agents. Different tasks may require agents with specific skills for performing them [6]. In [1] LEP was presented as a problem with homogeneous agents, i.e., all police units had the same set of skills. However, we argue that LEP can be a perfect example for a scenario as described above where agents may have different skills, e.g., police officers that specialize in defusing bombs, that have dogs that can detect drugs etc. Thus, we start the discussion of task allocation in heterogeneous multi agents scenarios by presenting the extension of LEP for such scenarios. Afterwards, we present the enhanced FMC_TA algorithm that has been adjusted for solving problems with heterogeneous agents.

4.1 LEP with Heterogeneous Agents

The extended LEP model (LEP^H) includes a set of k unique skills S. Each agent a_i has a subset $S_i \in S$ of skills.

In LEP^H, for each task v_j we redefine the term *required workload* to be a set W_j, where each member of such a set w_j^s specifies the workload of a specific skill s that should be applied to v_j in the combined effort for completing this task. Thus, v_j is completed only if all the work specified by members in W_j has been performed by agents that posses the required skills. Formally, the *total workload* for completing the task is $w_j = \sum_{W_j} w_j^s$.

The allocation specifies for an agent, not only what tasks to perform but also what skills to use. If the agent is required to use more than one skill in performing the task, it must do so sequentially. If multiple agents are performing different skills, they can do so concurrently. Formally, an allocation of tasks to agents in LEP^H is denoted by an $n \times m \times k$ matrix X where entry x_{ijs} is the fraction of task v_j that is assigned to agent a_i, utilizing the skill s. The agents' schedules must include, besides the task being performed and the start and end time, the skill the agent is utilizing, i.e., each member of σ^i includes (v_j, s, t, t') specifying the task, the skill being utilized, and the start and end time for applying skill s on task v_j by this agent, respectively.

In LEP the utility agents derive for performing task v_j depends on the number of agents that work simultaneously on the task. In LEP^H their might be additional inter skills constraints that require concurrency between police officers with different skills. Thus, *capability* function, is defined for a vector $\vec{q} \in \mathbb{N}^k$ that specifies the number of agents with each skill working concurrently on a task, i.e., The l'th entry in $\vec{q}$ represents the number of agents with skill s_l working on the task concurrently. We note, that each agent can be counted only once, i.e., it cannot utilize multiple skills simultaneously. The result of the function $Cap(v_j, \vec{q})$ is a vector $\vec{g}$ specifying for each skill, the utility derived by an agent performing it, taking under consideration the number of agents performing this skill and the inter skill constraints.

Let $d_{\vec{q}}^{v_j}$ be the time that $\vec{q}$ represents the set of agents working simultaneously on task v_j. Thus, $\frac{d_{\vec{q}}^{v_j}}{w(v_j)}$ is the relative portion of time that the set of agents specified by $\vec{q}$ are working on v_j. Denote by $\vec{Q}$ the set of all possible vectors $\vec{q}$. The utility derived by the agents for completing v_j is: $\sum_{\vec{q} \in \vec{Q}} \frac{d_{\vec{q}}^{v_j}}{w(v_j)} \sum_{l=1}^{k} q[l]g[l]$, where $q[l]$ and $g[l]$ are the l'th entry in vectors $\vec{q}$ and $\vec{g}$ respectively.

The definitions for the *soft deadline* function $\delta(t)$ remain the same. Thus, the discounted utility for performing task v_j with arrival time at time $\alpha(v_j)$ and which is initially handled at time t_{v_j} is: $U'(v_j) = \beta^{\gamma(t_{v_j} - \alpha(v_j))} \sum_{\vec{q} \in \vec{Q}} \frac{d_{\vec{q}}^{v_j}}{w(v_j)} \sum_{l=1}^{k} q[l]g[l]$

When a new task arrives, the *current task* (if any) being performed by agent a_i is denoted CT_i and the *current skill* that is used by a_i for CT_i is denoted CS_i. Agents can *interrupt* the performance of their current task. The *penalty* for task interruption, $\pi(v_j, \Delta w_j^{CS_i})$, which depends on the task v_j and the amount of work $\Delta w_j^{CS_i}$ for skill CS_i completed when the task is interrupted. The adjusted penalty for task v_j decreases exponentially with $\Delta w_j^{CS_i}$ to a minimum value:

$$\pi(v_j, \Delta w_j^{CS_i}) = \max\{I(v_j)c^{w_j^{CS_i} - \Delta w_j^{CS_i}}, \phi \cdot I(v_j)\},$$

where $c \in [0, 1)$ and $\phi > 0$ are constants and $\phi I(v_j)$ is the minimum penalty. The penalty is positive only if $w_j^{CS_i} - \Delta w_j^{CS_i} > 0$ and $CS_i \in S_i$, otherwise it equals to zero.

The total utility derived for performing v_j is thus

$$U(v_j) = U'(v_j) - \sum_{a_i : v_1^i \neq CT_i} \pi(CT_i, \Delta w_j^{CS_i})$$

4.2 FMC_TA with Heterogeneous Agents

FMC_TA for heterogeneous agents (FMC_TA^H) is a generalization of FMC_TA (introduced in Section 2) that can be used in scenarios with heterogeneous agents.

In FMC_TA^H each task v_j is represented as k subtasks v_{js}, where k is the number of skills in LEP^H. We denote R as a $3 - dimensional$ matrix of size $n \times m \times k$. Each entry r_{ijs} at time t represents the personal utility of agent a_i when immediately moving to handle subtask v_{js}. If agent a_i does not possess specialty s or task v_j does not require it, the value of entry r_{ijs} will be zero. The utility is constructed by optimistically ignoring the inter-task ordering constraints and assuming the maximum value for the capability function, as in FMC_TA.
Formally:

$$r_{ijs} = \beta^{\gamma(t + \rho(a_i, v_j))} \max_{\vec{q}}\{Cap(v_j, \vec{q})\} - \pi(CT_i, \Delta w_j^{CS_i})\}$$

where the penalty is omitted if $CT_i = v_j$.

FMC_TA^{H+} uses concave utility functions as presented in Section 3 as entries in matrix U (instead of matrix R). This matrix is used as an input for the Fisher Market Clearing mechanism. The utility function is $u_{ijk}(x_{ij}) = (h_{ijk}x_{ijk})^{\mu_j}$ where $\mu_j \in (0, 1]$ and $h_{ijk} = (r_{ijk})^{1/\mu_j}$.

In order to create the input for the FMC mechanism we transform the $3 - dimensional$ matrix U to $2 - dimensional$ matrix U' of size $n \times ms$. We solve the FMC problem by using the polynomial-time algorithm as described at [10]. The FMC mechanism produces an output of $2 - dimensional$ matrix X' that is transformed to a $3 - dimensional$ allocation matrix X, as is defined in the section above.

The second stage of FMC_TA^{H+}, where allocated tasks are scheduled for each agent to reflect the spatio-temporal inter-task and inter-agent constraints remains the same as in FMC_TA.

5. EXPERIMENTAL EVALUATION

In our experimental study we estimate the contribution of the two extensions we proposed for FMC_TA. First we evaluate the success of FMC_TA^+, the extension of FMC_TA that enables the use of concave personal utility functions as input to the FMC mechanism, by comparing it to the standard version that uses linear personal utility functions. Second we compare the success of FMC_TA^{H+} in solving LEP^Hs, by comparing it to Simulated

annealing, the algorithm that was found most successful among the competitors of FMC_TA in [1].

5.1 Evaluation of FMC_TA^+

In our first set of experiments, the experimental design follows the design in [1], which resembles a realistic LEP. The city was represented by a rectangular region of the Euclidean plane of size 6×6 kilometers, divided into 9 neighborhoods of size 2×2, each with a patrol task. The setup includes 8-hour shifts (as in real police departments), with 9 agents patrolling (one in each neighborhood) at the beginning of each shift. The number of tasks arriving (i.e., the *load*) in a shift varied between 20, 40, 60, 80 and 100 missions. Tasks arrived at a fixed rate and were distributed uniformly at random in the city. These included four types of events of decreasing importance $I(v) = 2400, 1600, 1200, 800$ from type 1 to type 4, respectively. Patrols had $I(v) = 500$. Event types were selected randomly according to the distribution of real event types provided by law enforcement authorities in our home city: $30\%, 40\%, 15\%, 15\%$ of events were of type 1 to 4, respectively. The workloads were drawn from exponential distributions with means $58, 55, 45, 37$ for events of type 1 to 4, respectively [1].

The Cap quality of task execution for each agent was assumed to improve up to a maximum number of agents Q_v:

$$Cap(v, q) = \min\{\frac{q}{Q_v} I(v), I(v)\}$$

where $Q_v = 3, 2, 1, 1$ for tasks of type 1 to 4, respectively. The discount function used was $\delta(v, t) = 0.9^t$ for all v.

In [1], the FMC_TA algorithm, using linear personal utility functions as input, was found to dominate state of the art general and specifically designed algorithms. Therefore, to avoid redundancy, we omit the comparisons we performed with other algorithms and present comparisons of different versions of FMC_TA^+ with standard FMC_TA.

In order to examine the influence of concave personal utility functions on the performance of FMC_TA^+ we implemented multiple versions of the algorithm. All versions used linear utility functions (i.e., concave utility functions with $\mu = 1$) for tasks of type 3 and 4, since these are simple tasks with low importance for which cooperation is not required. For tasks of type 1 and 2 we examined a set of values for $\mu \in \{1, 0.9, 0.6, 0.3\}$.

We also report the results of a version of FMC_TA^+ ($FMC\_TA^+_{cond}$) that applies to the conditions for cooperation, based on the analysis of the small scenario, as presented in Section 3.1. In detail, for every task of type 1 and 2 we checked the following two conditions:

1. That the distance to the closest task is smaller than 5 kilometers, i.e., $dt = 5$.
2. That the highest ratio between the utility of the task and any other active task is smaller than 0.5, i.e. $rt = 0.5$.

If at least one of the conditions applied, a concave utility function with $\mu = 0.9$ was used as an input for this task. Otherwise, a linear utility function was used.

Figure 4 presents the average percentage of shared tasks, of missions of type 1 or 2 as a function of shift load. Different curves represent the use of different values of the parameter μ in the concave utility function. As mentioned above, the curve that represents FMC_TA^+ using the concave utility function with $\mu = 1$ is equivalent standard FMC_TA using linear functions as input. Thus, in the rest of this section we refer to FMC_TA^+ with $\mu = 1$ as FMC_TA to avoid confusion. The $FMC\_TA^+_{cond}$ version of the algorithm used a concave function with $\mu = 0.9$ when the conditions applied and therefore is denoted by $\mu = 0.9+$ in the figure.

It is apparent that for low loads (20, 40 and 60 missions per shift) there is no change in the sharing percentage. However, for high loads (80 and 100 missions) the results indicate that sharing is more prevalent when $\mu < 1$. Moreover, in $FMC\_TA^+_{cond}$ agents share more tasks than in the other FMC_TA^+ versions.

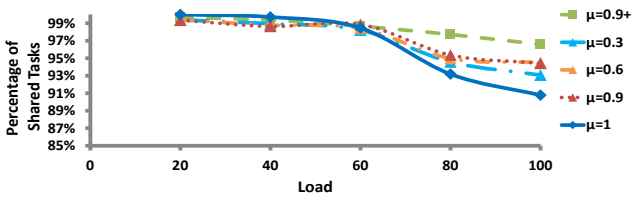

Figure 4: Cooperation as a function of shift load.

Figure 5 presents the average portion of time where three agents work simultaneously on a mission of type 1, as a function of shift load. All versions of FMC_TA^+ show a significant increase in simultaneous work time for all shift loads, in comparison with FMC_TA (for all comparisons $p\_value < 0.05$). $FMC\_TA^+_{cond}$ has an even higher percentage of shared time than FMC_TA^+ for all shift loads. For standard FMC_TA, the simultaneous working time between 3 agents starts at 50% at shift load 20 and decreases. In the busiest shift, with 100 missions, the sharing time is below 10%, i.e., the police units hardly cooperate. However, in $FMC\_TA^+_{cond}$, the agents work simultaneously on the same task

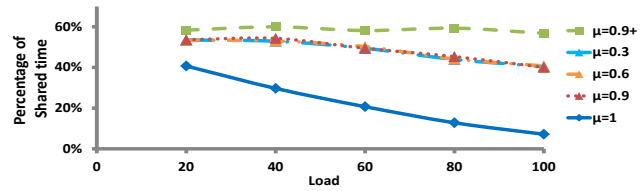

Figure 5: Cooperation time as a function of shift load.

Figure 6 and Figure 7 presents team utility and the average execution delay, receptively, as a function of shift load. The utility increases as shift load increases for all versions of the algorithm. However, the utility in the standard FMC_TA version grows more moderately when the shift load grows, than the utility of FMC_TA^+ and $FMC\_TA^+_{cond}$. The differences in utility between all FMC_TA^+ versions and FMC_TA are significant for high shift loads (80 and 100). In the shifts with the highest load (100) $FMC\_TA^+_{cond}$ achieves higher team utility than FMC_TA^+ but the difference is not significant.

The execution delay increased as the load of the shift increased, for all versions of the algorithms. Yet, the more concave the utility function used by FMC_TA^+ was, the higher was the execution delay. There are significant differences between the curves that represent FMC_TA with a linear utility function ($\mu = 1$) and FMC_TA^+ using concave utility functions with $\mu \in \{0.3, 0.6\}$, $p\_value < 0.05$.

These results are consistent with the results obtained for the level of sharing as presented in Figures 4 and 5 and the analytical results presented in Section 3.1 for the small scenario. The use of con-

cave functions in FMC_TA^{+} directly causes an increase in cooperation and as a result, leads to higher social welfare, albeit with higher execution delay. Nevertheless, when using personal utility functions with minor concavity (μ = 0.9), we get high team utility and minor execution delay.

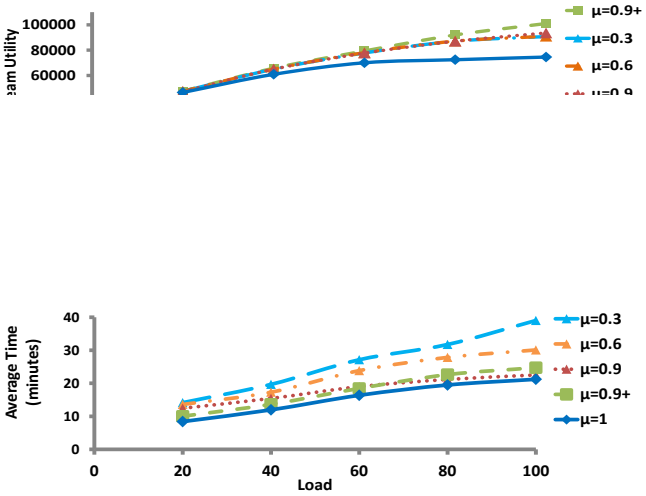

Figure 7: Execution delay as a function of shift load.

The results presented above show a clear advantage in the use of concave utility functions within FMC_TA. They show an increase in the general rate of sharing tasks and a significant increase in the sharing of the most important tasks by more than two agents. As a result the team utility increases for all versions of concavity in which $\mu < 1$.

Interestingly, while the differences in the concavity of the function had very small effect on the quality of the allocations in terms of team utility (in some cases the smallest concavity, $\mu = 0.9$ produced the highest utility), the execution delay was affected very differently by the different μ values. While the execution delay of FMC_TA^{+} with $\mu = 0.9$ was almost similar to FMC_TA, the execution delay of the version with $\mu = 0.3$ was much higher. This indicates that a function with small concavity is enough to increase the cooperation of the agents to the desired level, and without deteriorating much the execution delay.

5.2 Evaluation of FMC_TA^{H+}

In order to evaluate the contribution of FMC_TA^{H} we extended the experimental setup presented above to an LEP^{H}. We used two setups in our experiments. The first included two skills $s_1, s_2 \in S$. The sets of skills of six agents $a_1, ..., a_6$ included only s_1 and the sets of skills of the remaining three agents a_7, a_8 and a_9 included only s_2. As in the previous experiment the setup included four types of events of decreasing importance $I(v) = 2400, 1600, 1200, 800$ from type 1 to type 4, respectively and the workloads were drawn from the same exponential distributions.

The workload for missions of different types is described using vectors of size two, that represent the required workload for each skill. The workload for missions of type 1 was $\vec{w}(v_j) = (2w_j/3, w_j/3)$, for missions of type 2 was $\vec{w}(v) = (w_j/2, w_j/2)$ and for missions of types 3 and 4 the workload was $\vec{w}(v_j) = (w_j)$.

The Cap function varied for different types of tasks. The execution quality was monotonically non-decreasing as more agents perform the task concurrently, up to a specific number of agents for each of the skills. When this specific number of agents for some skill is reached, adding additional agents with this skill can no longer improve execution quality. For tasks of type 3 and 4 the Cap function was:

$$Cap(v_j, \vec{q}) = \begin{cases} I(v) & \text{if } q_1 \geq 1, q_2 \in \mathbb{N} \\ 0 & otherwise \end{cases}$$

For tasks of type 2 the Cap function was:

$$Cap(v_j, \vec{q}) = \begin{cases} I(v) & \text{if } q_1 \geq 1, q_2 \geq 1 \\ I(v)/3 & \text{if } q_1 \geq 1, q_2 = 0 \vee q_1 = 0, q_2 \geq 1 \\ 0 & \text{if } q_1 = 0, q_2 = 0 \end{cases}$$

For tasks of type 1 the Cap function was:

$$Cap(v_j, \vec{q}) = \begin{cases} I(v) & \text{if } q_1 \geq 2, q_2 \geq 1 \\ I(v)/2 & \text{if } q_1 \geq 2, q_2 = 0 \vee q_1 = 1, q_2 \geq 1 \\ I(v)/4 & \text{if } q_1 = 1, q_2 = 0 \vee q_1 = 0, q_2 = 1 \\ 0 & \text{if } q_1 = 0, q_2 = 0 \end{cases}$$

The discount function was $\delta(v, t) = 0.95^{t}$ for any v.

We examined the FMC_TA^{H} algorithm with only linear utility function as input to FMC and the FMC_TA^{H+} algorithm where a linear utility function was used (i.e., concave utility function with $\mu = 1$) for tasks of type 3 and 4, and for tasks of type 1 and 2 a concave utility function with $\mu = 0.9$ was used.

We compared the performance of FMC_TA^{H} with the performance of Simulated annealing (SA). SA was selected not only because it was found to dominate all other algorithms that FMC_TA was compared with in [1], but it was also the only other algorithm that allowed agents to share tasks efficiently (although not as efficient as FMC_TA).

Figure 8 presents the team utility (SW) as a function of shift load. As expected, for the lower loads (20, 40, 60) the utility increases with the load and all algorithms produce the same team utility. However, for the high loads (80, 100) team utility in FMC_TA^{H} and FMC_TA^{H+} continues to increase while in SA it decreases. For the high loads (80, 100) the difference in utility between FMC_TA^{H+} and SA is significant, $p\_value < 0.05$. In addition for the highest load(100) the difference in utility between FMC_TA^{H+} and FMC_TA^{H} is significant as well.

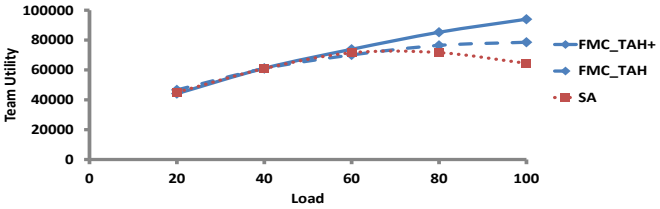

Figure 8: Team utility as a function of shift load in LEP^{H}.

Figure 9 presents the percentage of shared tasks of type 1 (upper graph) and type 2 (lower graph). For shifts with smaller loads (20, 40, 60) when both algorithms were used, for both types of mission, nearly 100% of the tasks were shared. However, for shifts with high loads, the agents that used FMC_TA^{H} or FMC_TA^{H+} still shared close to 100% of the tasks while in SA, the rate of shared tasks decreased. Specifically, at the highest shift load, 100, the sharing rate for type 1 missions decreased below 70% when SA was used.

Figure 10 presents the average execution delay as a function of the load. Time increases with shift loads, for all three algo-

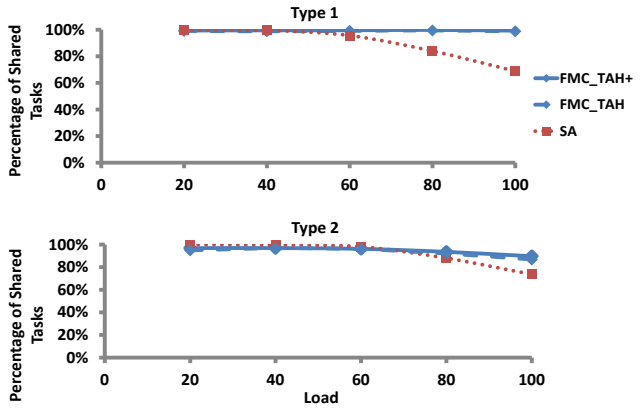

Figure 9: Cooperation rate as a function of shift load in LEP^H.

rithms. However, in shifts with the highest load, FMC_TA^H and FMC_TA^{H+} have a significantly lower average execution delay.

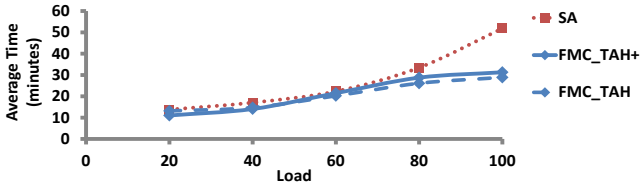

Figure 10: Execution delay as a function of shift load in LEP^H.

Figure 11 presents the average rate of abandoned tasks (tasks that agents stopped their execution before completion) as a function of the load. Apparently, the percentage of abandoned tasks increases with shift load. FMC_TA^H and FMC_TA^{H+} have significantly lower percentage of tasks that were not completed compared to SA, for all types of shifts.

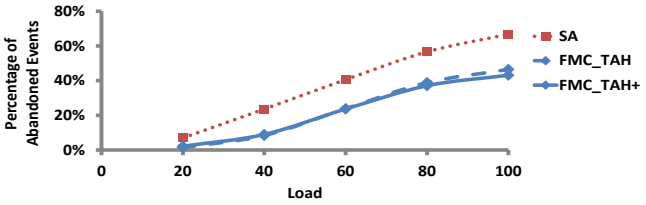

Figure 11: Abandonment rate as a function of shift load in LEP^H.

The second setup was larger and included five skills $s_1, \ldots, s_5 \in S$. We increased the size of the city to 10×10 kilometers, divided into 25 neighborhoods of size 2×2. The number of agents was increased to 25 and each agent had two skills that were chosen randomly (uniformly), the first between s_1 and s_2, and the second between s_3, s_4 and s_5.

We further increased the shift loads, with respect the the higher number of agents, to include 56, 111, 167, 222 and 287 missions. Tasks arrived at a fixed rate and were distributed uniformly at random in the city. As in the previous setups there are four types of tasks with the same importance and the same exponential distributions for the workload. Tasks of type 3 and 4 require only one skill, s_1 and s_2 respectively. Task of type 2 require two skills, s_2 and a randomly chosen skill from the set $\{s_3, s_4, s_5\}$. Tasks of type 1 require two skills, s_1 and a randomly chosen skill from the set $\{s_3, s_4, s_5\}$. The workload division between the different skills, the Cap functions and the discount function remain the same, as described above for the smaller setup.

The results for the larger problem setup in general were consistent with the results obtained for the smaller setup. Figure 12 presents the average team utility as the function of the load. FMC_TA^H and FMC_TA^{H+} dominate over SA in the high loads (222,278). The difference in team utility between FMC_TA^{H+} and SA for the highest load (278) remains the same, 32%. However, the difference between FMC_TA^H and FMC_TA^{H+} for the highest load is not significant in the large problem setup (unlike in the small setup).

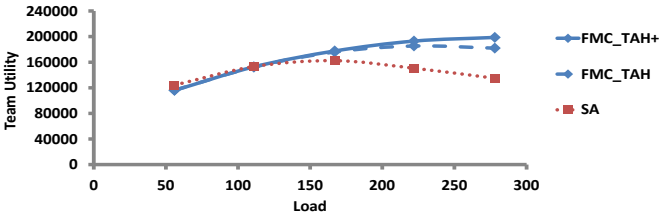

Figure 12: Team utility as a function of shift load in LEP^H (larger setup).

As mentioned above, the results for other metrics: average execution delay, average sharing rate and average abandonment rate, are similar to the results obtained for the small setup and were omitted for lack of space.

6. CONCLUSION

FMC_TA was recently proposed for solving dynamic task allocation problems in general, and in particular for solving LEPs. In [1], FMC_TA was found to outperform benchmark algorithms in both centralized and distributed settings. Despite its success, the scenarios in which it could be applied were limited to scenarios with homogeneous agents, and scenarios where sharing of tasks is not strictly preferred over single handling of tasks.

In this paper we enhanced FMC_TA by introducing the use of concave personal utility functions as the input to the FMC mechanism and by extending it to apply to teams of heterogeneous agents, i.e., agents with different skills.

By using concave personal utility functions we incentivized cooperation between agents. We analyzed the different cases in a small LEP with two police units and two tasks, and proved that in this small scenario, cooperation increases the team utility but at the same time enlarges the execution delay. Our empirical results demonstrate that small concavity of the function is enough to encourage a desired level of cooperation, which can be achieved with a minor effect on the execution delay. Based on the properties we proved for the small case scenario, we designing a heuristic for deciding for which tasks to use concave utility functions as input to the FMC mechanism. The version of the algorithm using this heuristic generated more cooperation among agents and resulted in better performance in terms of team utility and execution delay.

We further extended the LEP model presented in [1] to represent scenarios in which specific skills of agents are required in order to perform tasks and extended the FMC_TA algorithm to handle such scenarios, by extending the Fisher market to include products for every skill required for every task (and not just product per task as in FMC_TA). Our experiments reveled a large advantage of FMC_TA^{H+} over SA.

REFERENCES

[1] S. Amador, S. Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *AAAI*, 2014.

[2] W. C. Brainard, H. E. Scarf, et al. *How to compute equilibrium prices in 1891*. 2000.

[3] A. E. Clark and A. J. Oswald. Comparison-concave utility and following behaviour in social and economic settings. *Journal of Public Economics*, 70(1):133–155, 1998.

[4] D. Gale. *The Theory of Linear Economic Models*. McGraw-Hill, 1960.

[5] E. G. Jones, M. B. Dias, and A. Stentz. Learning-enhanced market-based task allocation for oversubscribed domains. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, November 2007.

[6] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in RoboCup Rescue. *The Computer Journal*, 53(9):1447–1461, 2010.

[7] S. D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings. Coalition formation with spatial and temporal constraints. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 1181–1188, Richland, SC, 2010.

[8] C. R. Reeves. Modern heuristic techniques for combinatorial problems, 1993.

[9] J. H. Reijnierse and J. A. M. Potters. On finding an envy-free Pareto-optimal division. *Mathematical Programming*, 83:291–311, 1998.

[10] L. Zhang. Proportional response dynamics in the Fisher market. *Theoretical Computer Science*, 412(24):2691–2698, 2011.

[11] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraints optimization problems in sensor networks. *Artificial Intelligence*, 161:1-2:55–88, January 2005.