Market Clearing–based Dynamic Multi-agent Task Allocation

SOFIA AMADOR NELKE, HIT Holon Institute of Technology
STEVEN OKAMOTO, Google
ROIE ZIVAN, Ben-Gurion University of the Negev

Realistic multi-agent team applications often feature dynamic environments with soft deadlines that penalize late execution of tasks. This puts a premium on quickly allocating tasks to agents. However, when such problems include temporal and spatial constraints that require tasks to be executed sequentially by agents, they are NP-hard, and thus are commonly solved using general and specifically designed incomplete heuristic algorithms.

We propose FMC_TA, a novel such incomplete task allocation algorithm that allows tasks to be easily sequenced to yield high-quality solutions. FMC_TA first finds allocations that are fair (envy-free), balancing the load and sharing important tasks among agents, and efficient (Pareto optimal) in a simplified version of the problem. It computes such allocations in polynomial or pseudo-polynomial time (centrally or distributedly, respectively) using a Fisher market with agents as buyers and tasks as goods. It then heuristically schedules the allocations, taking into account inter-agent constraints on shared tasks.

We empirically compare our algorithm to state-of-the-art incomplete methods, both centralized and distributed, on law enforcement problems inspired by real police logs. We present a novel formalization of the law enforcement problem, which we use to perform our empirical study. The results show a clear advantage for FMC_TA in total utility and in measures in which law enforcement authorities measure their own performance. Besides problems with realistic properties, the algorithms were compared on synthetic problems in which we increased the size of different elements of the problem to investigate the algorithm's behavior when the problem scales. The domination of the proposed algorithm was found to be consistent.

CCS Concepts: • **Computing methodologies** → **Multi-agent planning**; *Multi-agent systems*; Planning under uncertainty; • **Applied computing** → *Multi-criterion optimization and decision-making*;

Additional Key Words and Phrases: Distributed Task Allocation, Multi agent system

ACM Reference format:
Sofia Amador Nelke, Steven Okamoto, and Roie Zivan. 2020. Market Clearing–based Dynamic Multi-agent Task Allocation. *ACM Trans. Intell. Syst. Technol.* 11, 1, Article 4 (January 2020), 25 pages.
https://doi.org/10.1145/3356467

This article extends our AAAI 2014 [1] paper by extending the presentation, adding examples, and investigating larger problem scenarios with different ratios between the size of the problem, the number of events that need to be handled, and the number of agents that can handle them.

Authors' addresses: S. A. Nelke, HIT Holon Institute of Technology, 52 Golomb Street, POB 305, Holon 5810201 Israel; email: sofiaa@hit.ac.il; S. Okamoto, Google, 805 N Highland Ave, Pittsburgh, PA 15206, USA; email: stevenokamoto@gmail.com; R. Zivan, Ben-Gurion University of the Negev, P.O.B. 653 Beer-Sheva 8410501 Israel; email: zivanr@post.bgu.ac.il.

2157-6904/2020/01-ART4 $15.00
https://doi.org/10.1145/3356467

1 INTRODUCTION

In law enforcement, police officers conduct routine patrols and respond to reported incidents (*tasks*). Each task has an *importance* ranging from low (e.g., noise complaint) to high (e.g., murder) and a *workload* indicating the amount of work that must be completed for the incident to be processed. Multiple officers may work together on especially important tasks, sharing the workload and improving response quality. Delays in arriving at the scene of an incident allow perpetrators to escape and situations to escalate. This is reflected by a *soft deadline* that decreases the utility derived from performing a task the later it is started. A task cannot be started until at least one of the assigned officers arrives at the location of the task, and this depends both on the relative locations of officers and tasks as well as on the sequence of other tasks that the officers are scheduled to perform. Finally, because new incidents may be reported at any time, allocations must react to dynamic changes with officers sometimes interrupting execution of low-priority tasks, even though this negatively impacts the performance of the interrupted task. We term the problem of finding allocations that maximize team utility over time the *Law Enforcement Problem (LEP).*

In this article, we make three major contributions. First, we formalize LEP, which, despite its name, is also applicable to realistic multi-agent applications, e.g., disaster response, multi-robot allocation, and military teams (see Reference [23] for a taxonomy of task allocation problems). These domains share four properties that make task allocation extremely challenging: task heterogeneity, cooperative task execution, spatial and temporal constraints, and a dynamic environment. This combination of features has not previously been studied. Cooperative task execution imposes inter-agent constraints that increase the utility for shared tasks while decreasing the time a single agent must spend on the task. Utility also depends on the order in which tasks are executed (as a result of penalties for late execution of tasks). LEP includes additional challenging features, such as the option to interrupt a lower priority task for a more acute task, taking into consideration the cost for not completing the task, and the soft deadline representation of the cost for late execution of tasks [23]. Although LEP is NP-hard (the traveling salesman problem is a special case with a single agent), dynamism and soft deadlines require that solutions be found quickly before the environment changes (e.g., a new task arrives) or utility is lost due to agents' late arrivals to tasks.

Our second contribution is FMC_TA, a novel algorithm that requires worst-case polynomial and pseudo-polynomial time in centralized and distributed settings, respectively. Instead of directly searching the space of execution schedules, FMC_TA first finds an allocation by using a simplified problem model that ignores spatial-temporal and inter-agent constraints. Next, it schedules each agent's tasks based on the full problem model. The key challenge is to choose an allocation that can be scheduled to yield a high-quality solution.

FMC_TA borrows two concepts from non-cooperative multi-agent systems, *envy-freeness* and *Pareto optimality*, to do this. An allocation is envy-free if no agent prefers the allocation of another agent to its own, and is Pareto optimal if the team utility (i.e., *social welfare*) cannot be increased without any agent suffering a decrease in individual utility [33]. Although agents in LEP are cooperative, we hypothesize that envy-freeness avoids long delays by balancing the task load among agents [15], while Pareto optimality helps to allocate high importance tasks to sets of agents with high capability. Thus, allocations with both properties will be able to be sequenced to yield high-quality solutions.

An envy-free, Pareto optimal allocation can be found by using the Fisher market clearing (FMC) model [3, 7, 13, 33, 42]. FMC_TA computes an FMC allocation by modeling agents as buyers, tasks as goods, and simplified utilities as preferences. Individual tasks, especially important ones, may be divided between multiple agents who share the workload.

FMC_TA then schedules the allocation, taking into account inter-task and inter-agent constraints. The tasks allocated to each agent are prioritized by importance or/and maximal derived utility. These sequences are then modified based on inter-agent constraints. A unit that arrives to a location of a task, which it shares with other units, before the other units arrive may start handling the task. However, the utility it derives from performing the task is less than the utility the units will derive from performing the task together collaboratively. Therefore, agents do not benefit from arriving earlier than the agent that arrives the latest to a shared task. Hence, for each agent that is scheduled to arrive early to a shared task, we move its next scheduled task before the shared task, if this does not delay the execution of the shared task.[1]

Our third contribution is an empirical evaluation of FMC_TA, in which we compare its performance with the performance of alternative task allocation algorithms, and general optimization algorithms in both centralized and distributed settings. We evaluated the algorithms on realistic problems based on actual log files provided by the police force in our home city. FMC_TA achieves higher team utility and also surpasses competitors on other measurements used by real-world law enforcement authorities to evaluate performance, in comparison with state-of-the-art incomplete algorithms, both centralized and distributed [16, 31, 32, 44]. We further evaluated the proposed algorithm on problems with different characteristics. Specifically, we conducted experiments with a larger number of police units, larger cities, and an increasing event rate. Our proposed algorithm was found to dominate state-of-the-art heuristics on larger settings.[2]

2 RELATED WORK

Task allocation can be divided into two main categories: weak self-organization mechanisms, where a central agent has to allocate tasks to other agents in the mechanism; or strong self-organization mechanisms, in which the allocation is performed distributively by the agents to whom the tasks are allocated [40]. There are diverse algorithms, centralized and distributed, that are intended to solve task allocation problems [19, 25, 31, 35, 37]. Some of them are market-based mechanisms that have been applied to resource allocation on computers [18, 39], patient scheduling in hospital [26], multi-agent [6, 12, 14, 16, 22], and multi-robot coordination [4, 8, 17, 21, 45]. In addition, there are auction-based mechanisms that solve supplementary problems, e.g., multi-agent patrolling [9] and area exploring [38]. Like in our algorithm, these mechanisms were applied to task allocation by having the computers, agents, or robots take the role of the buyers in the market and having tasks or resources as products. We present the most relevant and state-of-the-art approaches for solving task allocation that can be used as a benchmark to our approach.

A work that addresses the same problem domain as our own, but uses a completely different algorithmic approach is Reference [31]. It presents a coalition formation with spatial and temporal constraints problem (CFSTP). Agents form coalitions to jointly work on tasks with spatial constraints and deadlines. Agents sharing a task in a CFSTP work at non-additive rates, and the objective is to maximize the number of tasks completed before hard deadlines. CFSTP was originally solved using Coalition Formation with Look-Ahead (CFLA) [31], which applies two heuristics: allocating the smallest possible coalition and maximizing the number of other tasks that can

[1]Our description of this contribution was extended, in comparison to the conference paper [1], including an example in which we analyze the bounds on the difference from the optimal solution and the benefit of cooperation when solving a small scenario.

[2]Our empirical evaluation is much more comprehensive in comparison with the conference paper [1]. In the conference version, the experiments only included a realistic scenario, following the police data regarding the number of police units in the city and in each neighborhood. Here, to investigate the properties of the algorithm, we performed parameter analysis and evaluated the algorithm performance when the problem scales.

be completed before the deadline in the next time step. CFLA's heuristics are not helpful when tasks can be performed by individuals (albeit with lower quality than when performed by groups), deadlines are soft, and tasks have different inherent importance. CFSTP was later solved with the distributed Max-sum DCOP algorithm [19, 30], but the soft deadlines in LEP result in constraints of high arity that Max-sum requires exponential time to solve [11].

A work that uses a different mechanism to accomplish a similar goal is Reference [37]. It presents a distributed market protocol for allocating tasks to agents that contend for scarce resources. Agents trade tasks at prices determined by an auction protocol.

In Reference [16], a market-based mechanism was applied to a firefighting problem, a realistic domain with many similar features to the law enforcement problem. In that work, though events (fires) were assigned to firefighting units one-by-one according to the order in which they were discovered, each fire had an auctioneer, and agents computed the change in utility for being allocated a task, using either an optimal or near-optimal ordering, and submitted this change as a bid. The auctioneer greedily and irrevocably allocated the task to the agent with the highest bid. Once a fire had been assigned to an agent, the agent was responsible for it until it was extinguished. This is in contrast to the approach we took in this research where all tasks that are not complete when we run the mechanism are considered for reallocation, even if they were already assigned to agents in previous rounds.

There are a number of studies that aim to distributed auction-based mechanisms, based on consensus [5, 28]. They aim at allowing to implement auction-based allocation in distributed scenario, without distributing all the information of the problem to all the agents. In our experimental study, we compare our proposed algorithm with centralized auction-based task allocation algorithms and demonstrate the advantage of our approach.

Considering reassignment of allocated tasks increases the problem's complexity. However, the tractability of the market-clearing algorithm we use makes it possible. Moreover, our mechanism can allocate complex events, or events with high importance, to a number of agents that collaborate to handle them. This work, like ours, can be implemented centrally and distributively.

In our algorithm, we reduce utility for late execution of tasks and when agents do not complete the execution of tasks. We are not the first to use fines to incentivize agents, e.g., Reference [24] have used fines for late execution while Reference [27] used fines for reallocation before completing tasks. However, these studies, contrary to ours, assume that there are more agents than tasks. Thus, the effect of this use of fines is expected to be different.

General optimization metaheuristics such as simulated annealing [32, 34, 36] and hill climbing with random restarts [29] have also been used for task allocation. We found simulated annealing to dominate other metaheuristics for LEP, consistent with previous task allocation studies [34]. Apparently, modeling LEP as a standard optimization problem results in allowing general off-the-shelf optimization algorithms to perform well and produce high-quality results as they do in other domains. General distributed local search algorithms can also be used for this purpose in distributed settings [10, 41, 44].

Market-clearing models have been studied in economics for more than a century, starting with the work of Fisher and Walras from the end of the 19th century. Later work by Eisenberg and Gale [13] proposed a convex program for solving the Fisher model. The Arrow-Debreu model, in contrast to Walras's preliminary work, is a market exchange model that guarantees the existence of a solution under some assumptions. More recent work has developed a polynomial-time centralized algorithm [7] and a pseudo-polynomial time distributed algorithm [42]. Our work is, to the best of our knowledge, the first attempt to apply a market-clearing approach to realistic dynamic task allocation problems.

3 LAW ENFORCEMENT PROBLEM

In formalizing LEP, we first consider the simpler static problem before turning to the full, dynamic version.

3.1 Static Problem

In the static *LEP* there are n cooperative, homogeneous 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. The time it takes to travel between two locations is given by the function $\rho : L \times L \to [0, +\infty)$. With slight abuse of notation, we write $\rho(a_i, v_j)$ or $\rho(v_j, v_{j'})$ to denote the travel times between locations of an agent and a task or between locations of two tasks, respectively.

There are two kinds of tasks: *patrols* of neighborhoods and *events* that require a police response. Each task v_j has an *importance* $I(v_j) > 0$; in general, patrols are less important than events and thus, when events occur, agents leave their patrol task and attend the events. Events also have a *workload* $w(v_j)$ specifying how much work (in time units) must be performed to complete the task. Patrols have no workload but are ongoing tasks that are never completed. Agents derive positive utility at a constant rate while patrolling; hence, when they are not assigned a specific task, they have an incentive to patrol.

An allocation of tasks to agents is denoted by the $n \times m$ matrix X where entry x_{ij} is the fraction of task v_j that is assigned to a_i. Agents can only perform a single task at a time so the M_i tasks allocated to agent a_i must be scheduled, i.e., $\sigma^i = (v_{s_1^i}, t_1^i, {t_1^i}'), \ldots, (v_{s_{M_i}^i}, t_{M_i}^i, {t_{M_i}^i}')$ is a sequence of M_i triples of the form $(v_{s_k^i}, t_k^i, {t_k^i}')$, where $v_{s_k^i}$ is the task performed from time t_k^i to ${t_k^i}'$. The spatial-temporal constraints require that the time spent on each task must equal a_i's assigned share of the workload and that agents must have sufficient time to move between tasks, because they can only perform tasks at their current location:

$$ {t_k^i}' - t_k^i = x_{is_k^i} w\left(v_{s_k^i}\right) \qquad 1 \le k \le M_i \tag{1} $$

$$ t_{k+1} - t_k' \ge \rho\left(v_{s_k^i}, v_{s_{k+1}^i}\right) \qquad 1 \le k \le M_i. \tag{2} $$

We denote by t_{v_j} the first time that one of the allocated police units arrives at the scene: for all $v_j \in V$ there exists t_{v_j} such that for all $a_i \in A$, $v_j = v_{s_{k_i}^i} \Rightarrow t_{v_j} = \min\{t_{k_i}\}$.

Multiple agents can share a single event (e.g., officers interviewing different witnesses). At least one of the agents sharing a task must be present before task execution can begin, and the amount of time an agent must spend on the task is equal to its allocated fraction of the workload $x_{ij}w_j$; these portions may be of different sizes.

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, $Cap(v_j, q)$. This can represent minimum required or maximum allowed numbers of agents by setting capability to 0 for fewer agents or by not increasing the value of $Cap(v_j, q)$ when more than the maximum number of required agents share a task, as well as changes in execution quality due to synergies or coordination costs.

The utility derived for completing task v_j starting at time t_{v_j} depends on the capability of the agents performing the task and the *soft deadline* function $\delta(v_j, t) : V \times [0, +\infty) \to (0, 1]$, which is monotonically non-increasing in t. The decision to reduce utility as a result of a delay at the beginning of a task handling is supported by real-world law enforcement settings, where conditions tend to worsen (e.g., perpetrators escape, confrontations escalate) until police officers arrive. In consultation with the police, we use an exponentially decaying soft deadline function for events, $\delta(v_j, t_{v_j}) = \beta^{\gamma t_{v_j}}$, where $\beta \in (0, 1]$ and $\gamma \ge 0$ are constants. For a patrol there is no deadline so $\delta(v_j, t_{v_j}) = 1$.

Let $d_q^{v_j}$ be the time that q agents are working together on mission v_j. Thus, $\frac{qd_q^{v_j}}{w(v_j)}$ is the relative part of the mission that is performed by q agents (as mentioned above, $w(v_j)$ is the total time of the mission). The total utility for completing v_j is

$$U(v_j) = \delta(v_j, t_{v_j}) \sum_{q=1}^{n} \frac{qd_q^{v_j}}{w(v_j)} Cap(v_j, q)$$

and the total team utility is $U(V) = \sum_{v_j \in V} U(v_j)$.

3.2 Dynamic Problem

In the dynamic problem, tasks arise over time. We denote the *arrival time* of a task v_j by $\alpha(v_j)$. Tasks can only be assigned after arrival. The soft deadline decreases the value of tasks after their arrival, so $\delta(v_j, t_{v_j}) = \beta^{\gamma(t_{v_j} - \alpha(v_j))}$.

Because task arrival is unpredictable, the dynamic problem is represented as a sequence of static problems, each instantiated when a new task arrives. When a new task arrives, the *current task* (if any) being performed by a_i is denoted CT_i. Agents can *interrupt* the performance of their current task. For example, an officer handling a (low importance) loitering complaint when a (high importance) murder is reported may be ordered to stop what he is doing and attend to the murder. Agents do not return to interrupted tasks.

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. In consultation with the police, we assume that the penalty for an 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. This is consistent with real police settings where the first few minutes are commonly critical and costly to interrupt. There is no penalty for interrupting a patrol, so $\pi(v_j, \Delta w) = 0$ in that case.

Letting $U'(v_j) = \delta(v_j, t_{v_j}) \sum_{q=1}^{n} \frac{qd_q^{v_j}}{w(v_j)} Cap(v_j, q)$, the utility of performing v_j in the dynamic problem is thus

$$U(v_j) = U'(v_j) - \sum_{a_i : v_{s_1}^i \neq CT_i} \pi(CT_i, \Delta w).$$

4 FMC-BASED TASK ALLOCATION

A Fisher market 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 in the market 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 [33].

4.1 Centralized Algorithm

FMC_TA represents agents and tasks as buyers and goods, respectively, and endows each agent with an equal amount of money.[3] We construct R by optimistically ignoring the inter-task ordering constraints and assuming the maximum value for the capability function. Specifically, we set entry

[3]The use of money is purely an internal mechanism of the allocation algorithm.

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 other agents:

$$r_{ij} = \delta(v_j, t+\rho(a_i, v_j)) \max_q \{Cap(v_j, q)\} - \pi(CT_i, \Delta w), \tag{3}$$

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

We find market-clearing prices using the polynomial-time algorithm of Devanur et al. [7], then produce the allocation matrix X as described by Reijnierse and Potters [33].

It is clear that R is not expressive enough to represent the complex team utility function in the formal *LEP* definition. However, in the simplified problem represented by R, the properties of the FMC allocation ensure that we achieve an efficient allocation that is balanced over the agents. Our experiments demonstrate that this results in higher team utility than directly maximizing the utility represented by R.

In the second stage of *FMC_TA*, we schedule the allocated tasks for each agent to reflect the spatio-temporal inter-task and inter-agent constraints. We construct an ordering of tasks allocated to each agent a_i by greedily prioritizing. Our investigation of the best priority upon which the tasks of agents should be scheduled included five alternatives. We analyze the difference between them and empirically compare between them in Section 5.2.

Ties are broken in favor of older tasks, imposing a unique ordering across agents that reduces task starvation, an important concern in law enforcement scenarios. Once the initial order is selected, each agent computes the initial schedule σ_i by setting t^i_k and $t^{i\prime}_k$ to satisfy Equations (1) and (2) at equality. This takes $O(m \log m)$ time.

We then update the start times to reflect the inter-agent constraints that shared task execution leads to better performance and consequently to higher social welfare. This delays shared tasks and any subsequent tasks, as reflected by the equations:

$$t_{v_j} = \max\left\{t^i_k \mid v_{s^i_k} = v_j \wedge x_{ij} > 0\right\}, \tag{4}$$

$$t^i_k = \max\left\{t^i_{k-1} + \rho(v_{s^i_{k-1}}, v_{s^i_k}), t_{s^i_k}\right\}, \tag{5}$$

which can be solved by monotonically iterating through the sorted times in the initial schedules in $O(m \log m)$ time.

We next see if the order of tasks in the individual schedules can be optimized by moving tasks delayed by shared tasks earlier in the order without further delaying shared tasks. For a shared task $v_{s^i_k}$ with $k > 1$ and non-shared task $v_{s^i_{k+1}}$ with $k > 1$, the non-shared task is moved before the shared task if $t_{s^i_{k-1}} + \rho(v_{s^i_{k-1}}, v_{s^i_{k+1}}) + x_{is^i_{k+1}} w(v_{s^i_{k+1}}) + \rho(v_{s^i_{k+1}}, v_{s^i_k}) \leq t_{s^i_k}$. If $k = 1$ (the shared task is first) at time t, then the non-shared task is moved if $t + x_{is^i_2} w(v_{s^i_2}) + 2\rho(a_i, v_{s^i_2}) \leq \tau_{s^i_1}$. If the schedules are changed, then the start times are again updated using Equations (4) and (5). Each task is considered once, requiring $O(m)$ time.

FMC_TA thus computes the FMC allocation and schedules tasks, i.e., its runtime complexity is in worst-case polynomial time.

4.2 Distributed Algorithm

LEP is a naturally distributed problem where agents (police officers) need to decide what tasks to perform in a dynamic environment. While in the section above, we assumed that all information is known to a centralized authority, a more natural representation for the problem is a distributed setting in which each agent is aware of the tasks that are located close to it and the agents perform a distributed algorithm to allocate the tasks among them. In more detail, in the distributed setting, each agent knows only about tasks within a certain *threshold* distance D, i.e., it has positive preferences for those tasks. We form a Fisher market where r_{ij} is set according to Equation (3) if $\rho(a_i, v_j) \leq D$ and 0 otherwise. Knowledge of R is distributed among the agents, such that each of them only knows its non-zero entries.

To compute a market-clearing solution, we use the distributed proportional response algorithm [42]. For each task there is a *seller* who computes the allocation for that task. The seller can be the closest agent to the task or a dedicated agent for the task. Agents iteratively submit bids to sellers of the tasks within their threshold and are in turn awarded provisional allocations, which they use to modify their bids in the next round. This process converges in pseudo-polynomial time to an ε-approximate market-clearing solution and allocation.

Agents compute their initial schedules in the same way as in the centralized approach, then update start times for shared tasks. For each task $v_j = v_{s^i_k}$ allocated to it, agent a_i maintains an estimated start time $t^i_{v_j}$, initialized to t^i_k. If v_j is a shared task, agent a_i then communicates $(t^i_{v_j}, v_j)$ to all other agents $a_{i'}$ with whom it shares the task.

Upon receiving a (t_{v_j}, v_j) message (where $v_j = v_{s^i_k}$), a_i computes the change in time $\theta = t_{v_j} - t^i_{v_j}$. If $\theta > 0$, then for all $k' \geq k$, a_i adds θ to $t^i_{k'}$ and $t^i_{k'}$, and transmits the new values $(t^i_{k'}, v_{s^i_{k'}})$ to all agents sharing $v_{s^i_{k'}}$.

This process terminates in polynomial time with $t_{v_j} = t^i_{v_j} = t^{i'}_{v_j}$ for all $a_i, a_{i'} \in A, v_j \in V$ and Equations (4) and (5) satisfied. Because there are no deadlocks, at least one time t^i_k will become permanently fixed in each synchronous communication cycle, and so the number of communication cycles is bounded from above by the total number of tasks in the schedules. Each schedule can contain at most m tasks and there are n schedules, so the number of communication cycles is bounded by mn. Running time of distributed *FMC_TA* is dominated by the pseudo-polynomial time to compute the FMC allocation.

4.3 Task Ordering

We detail the next five different approaches for agents to schedule the tasks assigned to them in the second phase of the algorithm. Each approach prioritizes the tasks by a different characteristic and aims towards optimizing according to a different metric.

- **Max BPB (Bang Per Buck)** Prioritizes tasks according to their utility to time unit ratio. The task with the highest ratio has the highest priority. For each agent a_i, we calculate the ratio by dividing the personal utility the agent derives from the task by the time that the agent handles the task v_j. The time is composed of travel time to the task and the actual time the agent performs the task. Formally, the BPB ratio of agent a_i for mission v_j is defined by: $\frac{r_{ij}x_{ij}}{w(v)x_{ij}+\rho(a_i,v_j)}$. The hypothesis that motivates this approach is that it will increase the team utility over time.
- **Max r_{ij}** Ordering according to the entries of the R matrix. Tasks with high r_{ij} values come first. The r_{ij} entry represents maximum group benefit from the task, when the number of agents executing the task is as required.
- **Max $r_{ij}x_{ij}$** Ordering according to the personal utility of the agent from a task. In contrast to Max r_{ij}, where the joint contribution of agents in handling a task is what determines the ordering, here, the personal contribution of the agent attending the task, with respect to the portion of the task allocated to the agent, counts.
- **Max $I(v)$ Min $\alpha(v)$** Tasks are ordered first by event importance $I(v_j)$, such that urgent events get handled first (e.g., murder or robbery). If more than one event of the same importance was allocated to an agent, then the agent will order the one that arrived earlier (smallest $\alpha(v_j)$) first.

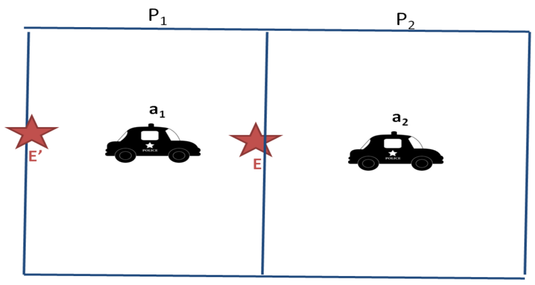

Fig. 1. Example.

- **Max $I(v)$ Max $\alpha(v)$** According to this ordering, the agent first handles the most important events and breaks ties in favor of the most updated event, from the events allocated to it. The rationale behind this ordering is that police officers can contribute more to recent events, where they can still influence the outcome of the event.

The ordering approaches are compared and the results are discussed in Section 4.3.

4.4 Example

The algorithm we present above for solving realistic task allocation takes into consideration only part of the problem's parameters. More specifically, the Fisher model can only consider unary preferences of agents and therefore binary constraints, such as the cost for a decision of a police unit to handle one event before the other, are ignored. One may wonder how do we expect the proposed method to compete with approximation methods that consider all of the problem's constraints and aim to achieve higher social welfare directly, e.g., Simulated annealing. To answer this question, besides explaining the intuitive motivation for our hypotheses, we present in this section a small scenario that includes two agents and the advantages and disadvantages for using our proposed algorithm for solving this scenario in comparison to a greedy algorithm. In the following section, we present empirical results that demonstrate the advantage of our algorithm in the general case.

Consider the example presented in Figure 1 that includes an artificial city with two patrol areas, p_1 and p_2, with equal size and importance, patrolled by two police units (agents), a_1 and a_2, respectively. We assume a discretized time horizon $t_0, t_1, \ldots$. Agents derive utility when they perform a task in a time unit, e.g., the utility agent a_i derives from patrolling for one time unit is $U_i(p_i), i \in \{1, 2\}$. For simplicity, we assume that the location of the agents when they are patrolling is in the center of the patrolling area (as depicted in Figure 1), although in fact, the exact location can be anywhere within the area assigned for patrol.

For an event E that arrives at time t_E, we denote by $w(E)$ the workload for event E in time units (i.e., the number of units required to handle the event). The utility for handling the event is $U(E)$. We denote by $\rho(a_i, E)$ the time that it takes agent a_i to reach event E.

A discount function for a delay of performing task E, $\delta(\rho(a_i, E))$, is a monotonically non-increasing function and $\delta(\rho(a_i, E) = 0) = 1$. For simplicity, we will use $\delta_i(E) = \delta(\rho(a_i, E))$.

The utility per time unit that agent a_i derives for handling event E is

$$U_i(E) = \frac{\delta_i(E) U(E) x_{i,E}}{2\rho(a_i, E) + w(E) x_{i,E}},$$

i.e., the discounted reward for performing an event is divided by the total time that the agent spent on the event (traveling time + handling). Obviously, an agent will decide to handle an event only if $U_i(E) > U_i(p_i)$.

The allocation that maximizes social welfare (the sum of utilities that the agents derive from an allocation) is denoted by OPT. Obviously, for a single event E, if $U_i(E) \leq U_i(p_i)$ when for each $i \in \{1, 2\}$ and $x_{i,E} \in (0, 1]$, then the optimal allocation for the team has $x_{1,E} = 0$ and $x_{2,E} = 0$, i.e., both agents avoid handling the event and keep patrolling. For any other case, where there is a single event E, and for at least one of the agents $U_i(E) \leq U_i(p_i)$, then $x_{1,E} + x_{2,E} = 1$.

If the single event E appears, as in Figure 1, close to the border of p_1 and p_2, such that $\rho(a_1, E) = \rho(a_2, E) - \epsilon$, where ϵ is a single time unit, then in the optimal allocation OPT, $x_{1,E} = 1$ and $x_{2,E} = 0$. This is because when $U_1(E) > U_1(p_1)$, the utility $U_1(E)$ is an increasing function of $x_{1,E}$; therefore, it reaches its maximal value when $x_{1,E} = 1$. Since $\delta_i(E)$ (the discount factor for $U_i(E)$) is monotonically decreasing in $\rho(a_i, E)$, obviously when $\rho(a_1, E) < \rho(a_2, E)$, OPT does not include $x_{1,E} = 0$ and $x_{2,E} = 1$. If $0 < x_{2,E} < 1$, then the social welfare value is approximately:

$$\delta_1(E)U(E) - [(4\rho(a_1, E) + w(E))U_1(p_1)].$$

This is smaller than

$$\delta_1(E)U(E) - (2\rho(a_1, E) + w(E))U_1(p_1),$$

which is the social welfare value when $x_{2,E} = 0$, since a_2 stays on patrol.

Thus, the social welfare value of the optimal allocation is

$$SW(OPT) = \delta_1(E)U(E) - (2\rho(a_1, E) + w(E))U_1(p_1)$$

when $\rho(a_1, E) \leq \rho(a_2, E)$. Denote by A_{fmc} the allocation that is produced by our proposed algorithm. When $U_2(E) > U_2(p_2)$ then in A_{fmc}: $x_{1,E} \geq x_{2,E} > 0$. Thus,

$$SW(OPT) - SW(A_{fmc}) = 2\rho(a_2, E)U_2(p_2).$$

However, assume that at time $t_{E'}$, $t_{E'} \geq t_E$ another event E' with the same importance as event E appears, with $\rho(a_1, E') = \rho(a_1, E)$ and $\rho(a_2, E') = 3\rho(a_1, E')$ (as depicted in Figure 1). We denote the distance from E to E' by $\rho(E, E')$ and assume that $\rho(E, E') = 2\rho(a_1, E')$. Then, A_{fmc} will allow the agents to complete event E faster and possibly reach E' faster, since in A_{fmc} event E will be handled in approximately half the time (to be accurate: $\frac{w(E)-\epsilon}{2} + \epsilon$). On the contrary, a greedy algorithm that allocates an event to the closest agent that is on patrol when it appears results in allocation A_g. Thus, we make the following observations:

(1) If E' appears later than the time it would take a_1 in A_g to handle E and return to its starting location, i.e., $k > 2\rho(a_1, E) + w(E)$ (where $t_{E'}$ is the time that E' appears), then E and E' are independent events. Notice that in both allocations, OPT and A_{fmc}, the handling of event E is completed before event E' appears. Thus, for both events the optimal allocation has $x_{1,E} = x_{1,E'} = 1$ and $x_{2,E} = x_{2,E'} = 0$.

(2) If E' appears at the same time as E both algorithms, then A_{fmc} and A_g would produce the allocation $x_{1,E} = 0, x_{2,E} = 1, x_{1,E'} = 1$ and $x_{2,E'} = 0$. This is obviously the optimal allocation with social welfare

$$SW(OPT) = 2\delta_i(E)U(E) - (4\rho(a_1, E) + w(E))U_1(p_1).$$

(3) If E' appears after E but before the handling of E would have been completed in either allocation, i.e.,

$$t_E < t_{E'} \leq \rho(a_1, E) + w(E)/2,$$

then if A_{fmc} was used, the position of the agents at $t_{E'}$ would be the location of E. In this case, allocating E' to a_1 and allowing a_2 to complete E would be most efficient. Notice that in A_g where $x_{1,E} = 1$, at $t_{E'}$ a_1 is located at E and a_2 at p_2. Thus, there are two options. Either a_1 leaves E or a_2 travels all the way to E'. Obviously, both cases are sub-optimal.

(4) If E' appears at a time such that if A_{fmc} was used, the agents would have completed handling E but if A_g was used, a_1 would not have completed it yet, i.e.,

$$2\rho(a_2, E) + w(E)/2 < t_{E'} < \rho(a_1, E) + w(E),$$

then in A_{fmc}, E' could be considered as a single event while in A_g, one of the two suboptimal options presented above must be used (either a_1 leaves E before completing it or a_2 travels the distance to E').

The example above, while very simple, indicates that there are cases where encouraging efficient cooperation between agents can be beneficial. This is true especially in cases where multiple tasks need to be handled concurrently.

5 EXPERIMENTAL EVALUATION

To evaluate the success of the proposed algorithm (FMC_TA), we compared it both to centralized and distributed state-of-the-art incomplete algorithms. Besides the standard team utility metric, we consulted with police officers and evaluated the algorithms according to specific metrics that are of interest to them: Average execution delay (the difference between the time a mission was added to the system and the time its execution started), average percentage of events whose execution was interrupted, average percentage of tasks that are shared (events that are intended for two agents or more and were allocated to more than one agent), average execution time of tasks, and the distribution of total shift time between the time in which agents are handling events, the time they spend patrolling, and time spent in transit on their way to events.

First, to find the best version of our proposed algorithm, we compared the performance of FMC_TA when using the different methods for agents to select the initial schedule of the tasks allocated to them as described in Section 4.3. Next, we compare FMC_TA to state-of-the-art incomplete algorithms, both centralized and distributed. Third, we analyze the resilience of the algorithm to different ratios between the number of tasks and the number of agents, in comparison to benchmark algorithms.

In all our experiments, we used variance analysis tests for estimating the overall statistical significance in the differences between group means. When the result of these tests was indeed significant, we used post hoc tests to confirm where the differences occurred between groups.

5.1 Experiment Design

As mentioned above, we compare *FMC_TA* to state-of-the-art incomplete algorithms, both centralized and distributed. Some of these algorithms consider only discrete allocations, e.g., Simulated annealing. Thus, we selected in advance the maximum number q of agents that can share a task and considered only allocations in which $x_{ij} = z/q$, where $z \in \{0, 1, \ldots, q\}$. For comparison, we applied these restrictions to *FMC_TA* by rounding the allocation matrix X. The algorithms were tested on independently generated random problems based on real police logs. The data we received from the police included log files of shifts, including event reports, responds, and reports on locations of units. Hence, we were able to analyze the rates of occurrences of different types of events and their probability in different areas of our home town. Unfortunately, secrecy restrictions prevent us from sharing this data. However, we used it to generate problems with realistic proportions of occurrences of events, the number of units, and the distances that units need to travel to events.

All results presented are averages of 100 simulated shifts. Unless stated otherwise, 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. We simulated eight-hour shifts as in real police departments, with nine agents patrolling (one in each neighborhood) at the beginning of

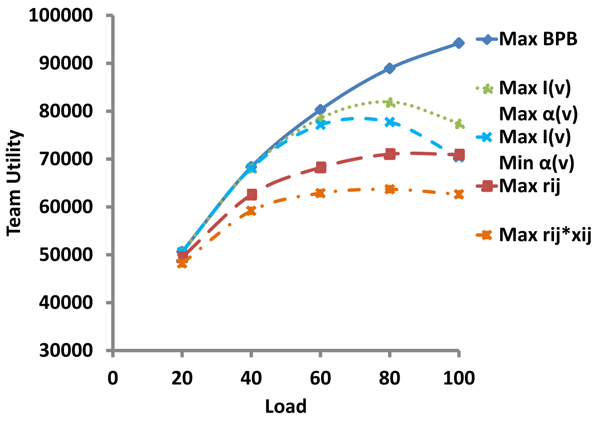

Fig. 2. Average team utility for varying heuristic orderings.

each shift. The number of tasks arriving (i.e., the *load*) could vary between shifts. We examined shifts with 20, 40, 60, 80, and 100 missions. Tasks arrived at a fixed rate and were distributed uniformly at random in the city.

There were four types of events of decreasing importance $I(v) = 2{,}400, 1{,}600, 1{,}200, 800$ 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. Based on estimates provided by the police, the workloads were drawn from exponential distributions with means 58, 55, 45, 37 for events of type 1 to 4, respectively.

We assumed Cap improved quality of task execution for each agent up to a maximum number of agents Q_v:

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

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.

5.2 Evaluation of Task Ordering

Section 4.3 details five different ordering methods. Figure 2 presents the team utility achieved by FMC_TA when using various scheduling methods for different loads of events. Figure 2 shows that using Max BPB, Max $I(v)$ Min $\alpha(v)$ and Max $I(v)$ Max $\alpha(v)$ monotonically increases team utility as the load increases. Max $I(v)$ Max $\alpha(v)$ and Max $I(v)$ Min $\alpha(v)$, however, exhibit a deterioration in team utility for higher loads.

Figure 3 presents the average execution delay of the tasks when using various scheduling methods for varying loads. Higher loads hurt performance. It can be seen that the execution delay increases as the load of the shift increases. There is no significant difference between the results for lower loads (20 and 60 events). However, for medium and high loads there is a gap. Max $I(v)$ Min $\alpha(v)$ has extremely high delays, and for shifts with 80 and 100 events per shift the difference between Max $I(v)$ Min $\alpha(v)$ and Max $I(v)$ Max $\alpha(v)$ (the second-worst method) is significant ($pvalue = 0.02$).

Team utility is directly affected by collaboration between units performing tasks, i.e., when units handle tasks simultaneously, they derive more utility. Figure 4 presents the percentage of shared tasks (out of tasks that require collaboration) as a function of load, when using various scheduling

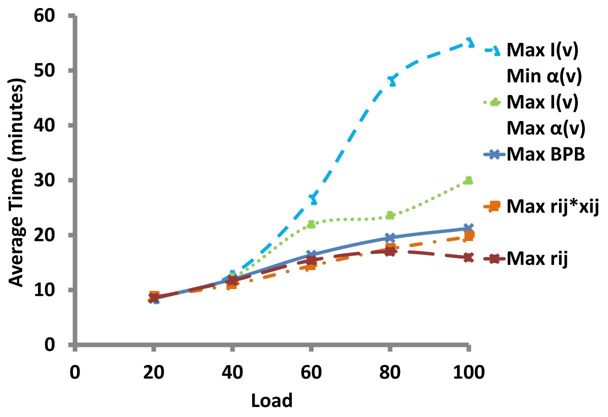

Fig. 3. Average execution delay for varying heuristic orderings.

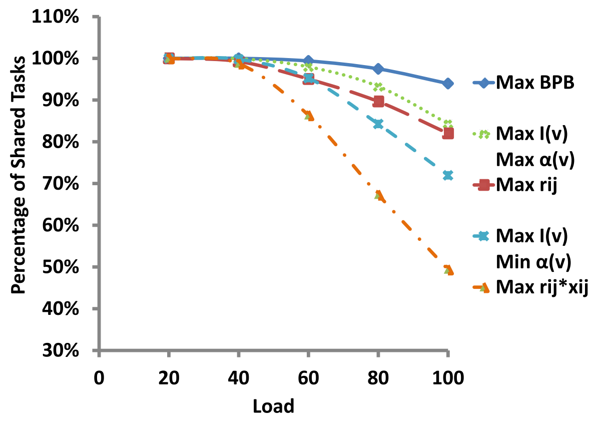

Fig. 4. Average percentage of shared tasks for varying heuristic orderings.

methods. For all methods, results show a decrease in sharing as load increases. High load shifts have more active tasks in the system simultaneously. To respond to as many tasks as possible, police units have to avoid collaboration. The Max BPB method maintains a high percentage of shared tasks. Even for the highest load, its sharing rate is above 93%. However, Max $r_{ij}x_{ij}$ drops to below 50% for the highest loads.

Figure 5 presents results of another important metric, percentage of abandoned (incomplete) events as a function of load. The figure shows that more events are abandoned as load increases. Figure 5 shows that methods Max r_{ij} and Max $r_{ij}x_{ij}$ have much higher percentages of interrupted events than other methods ($pvalue < 0.01$).

Figure 6 presents the division of time between attending tasks, moving, and patrolling during shifts. The results indicate that using the Max r_{ij} and Max $r_{ij}x_{ij}$ methods within *FMC_TA* cause agents to (in some cases) prefer patrolling over attending tasks.

5.2.1 Discussion. From a bird's eye view, there is no single dominant method and each has its own strengths. The Max BPB method dominates in team utility. For the highest load the difference between Max BPB and Max $I(v)$ Max $\alpha(v)$ (the second best) is significant ($pvalue = 0.02$). This does not come as a surprise, since Max BPB ordering explicitly maximizes the utility for each agent per time unit and thus implicitly maximizes team utility. Since team utility is strictly affected by

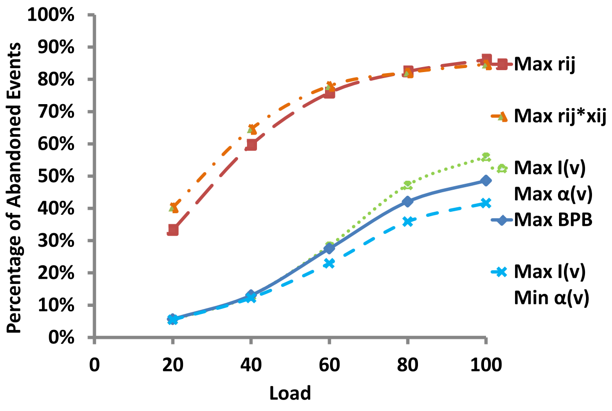

Fig. 5. Average percentage of abandoned events for varying heuristic orderings.

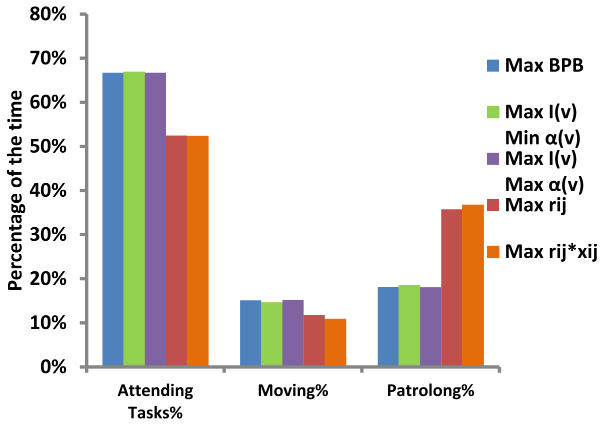

Fig. 6. Division of the time' 60 events per shift.

sharing, we see similar results for the sharing metric. For the rest of the metrics—average delay, percentage of interrupted events, and percentage of working time—Max BPB is not significantly better or worse than other approaches.

Max $I(v)$, Max $\alpha(v)$, and Max $I(v)$ Min $\alpha(v)$ prioritize by importance ($I(v)$) and time ($\alpha(v)$), two global characteristics, and do not take into consideration agents' personal utility r_{ij}. Both ordering methods generate similar schedules for agents that share tasks, which leads to a high average percentage of sharing. Similar schedules between two consecutive assignments lead to low percentage of interrupted events and high percentage of work time.

High percentage of shared tasks and low percentage of abandoned events, for Max $I(v)$ Max $\alpha(v)$ and Max $I(v)$ Min $\alpha(v)$, lead to high team utility. For low loads they produce results as high as Max BPB, the best performer (Figure 2), but they exhibit a drop for higher loads. A possible explanation is that as load increases, there are more important events and according to these scheduling methods the agents spend most of the task attending time in attending the most important ones, even if these are far away or their utility has decreased as a result of obsolescence. As a result, agents neglect the less important events and this leads to a decline in team utility. Another manifestation of this can be seen in the execution delay metric (Figure 3). According to this metric, Max $I(v)$ Min $\alpha(v)$ has significantly worse results in comparison with all methods.

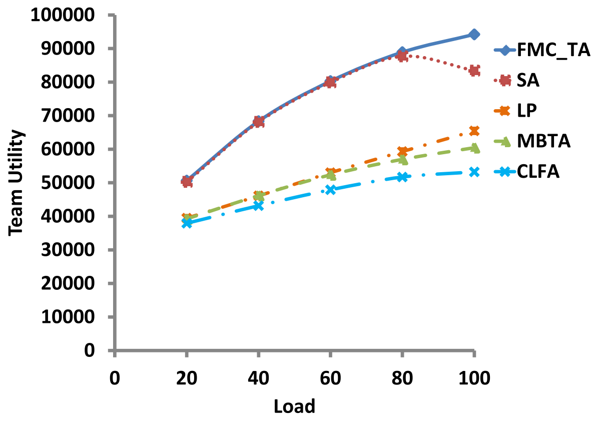

Fig. 7. Accumulated team utility for centralized algorithms.

Max r_{ij} ordering and Max $r_{ij}x_{ij}$ ordering have good results in average execution delay, because both orderings are based on personal utility r_{ij}, which is based on direct utility from task execution and increasing penalty for distance. Therefore, new and close events are prioritized. However, for high load shifts there is a trade-off between the arrival time metric and percentage of interrupted events. Attended events are abandoned for handling of new arriving events. This causes a high percentage of interrupted events, which then leads to lower working time and higher travel time. Another possible explanation is that the entry r_{ij} decreases as time passes, so according to these two methods, units abandon non-recent tasks and prefer patrolling.

As described in Section 4.3, there are trade-offs between different metrics. If only one measure was important for the police, then it would have been easier to choose, but this is not the case. The team utility aims at balancing all metrics. Therefore, in the rest of this section, we use the Max BPB ordering, which dominates according to team utility.

5.3 Evaluation of Centralized Algorithms

We compared the centralized version of *FMC_TA* to three state-of-the-art incomplete algorithms: Simulated annealing (SA) [32], which explicitly optimizes team utility (SW). SA was rerun after each dynamic event, using the previous allocation as the starting allocation for the subsequent run. SA moves from one phase to the next by random reallocation of the tasks (a random rescheduling of a task for a random agent or a random exchange of two tasks between two random agents). Market-based task allocation (MBTA) [16] uses exponential-time complete search to produce optimal schedules (without sharing and task reallocation). According to the MBTA mechanism, every new incoming task is up for auction and is allocated to the agent whose utility it improves the most. CFLA$^+$ is a version of the CFLA task allocation algorithm [31] adapted for LEPs. In the look-ahead phase, CFLA$^+$ computes the maximum utility for pairs of tasks taking into account soft deadlines.

We also compared the results of *FMC_TA* with a baseline approach that uses the same input matrix as *FMC_TA*, only that it calculates allocations using a linear program (LP) that directly maximizes the utility represented by R. Differences in solution quality between *FMC_TA* and LP thus reflect the effects of envy-freeness and Pareto optimality in the simplified problem. Note that LP does not share tasks, because its constraint matrix is totally unimodular.

Figure 7 presents average team utility for each algorithm as a function of the load of the shift. *FMC_TA* dominates for loads varying from 20 to 100. SA generally does well as expected of a meta heuristic directly optimizing the team utility objective, but for the highest shift load, 100, SA

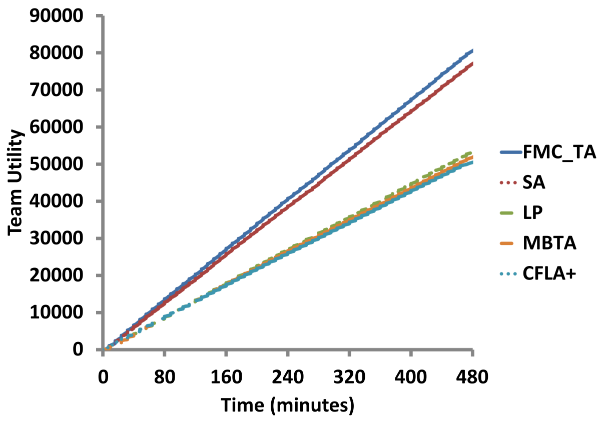

Fig. 8. Accumulated team utility, 60 events per shift for centralized algorithms.

apparently fails to deal with the high complexity of the problem, which leads to a decrease in team utility. The advantage of *FMC_TA* over SA, for shifts with 100 events, is significant ($pvalue = 0.04$). Although they are specialized task allocation algorithms, CFLA$^+$ and MBTA were not designed for the challenging combination of properties that LEP poses, and hence do not do as well. MBTA does poorly for low shift loads, possibly because it does not allow agents to share tasks; it was designed for oversubscribed domains where the benefits of quickly reaching any task outweigh possible benefits of cooperation.

Figure 8 presents the utility achieved over time for loads of 60 events. Agents accumulate utility at a steady rate using all algorithms and a partial F-test confirmed that agents using *FMC_TA* accumulate utility significantly faster than when using the other approaches ($pvalue < 0.01$). LP does worst, confirming that directly trying to maximize utility in the simplified problem is insufficient for achieving high utility in the original problem.

Figure 10 examines the delay before the execution of tasks begun as the function of the shift load. We note that there are no significant differences between the algorithms for lower loads, and the differences are observed only for the highest load (100). For very high loads, the system can become backlogged, with greater delay before tasks begin execution; MBTA and CFLA$^+$ are most affected by this, because they do not perform reallocation, thus they are less flexible and cannot cope with multiple appearances of important tasks.

Higher loads also mean higher dynamism, resulting in a higher rate of interruption. Figure 9 presents the average rate of abandoned tasks as a function of load. One may ask if interruptions are ever desirable, especially since the optimal ordering found by MBTA does not include them. However, this is a consequence of MBTA's inflexible allocation mechanism: A task can only be allocated once. CFLA$^+$ allows dynamic reallocation, but its one-step lookahead also precludes interruption. Explicitly disallowing interruption under *FMC_TA* resulted in poorer performance, but still better results than benchmark approaches, LP and SA ($pvalue = 0.02$).

Rapid response to incidents is highly valued by police departments, especially for more important events. Figure 11 presents the execution delay as a function of the event type for shifts with a load of 60. Agents using *FMC_TA* are especially responsive to high importance tasks. The delay increases as the importance decreases. SA has the same trend but with a significantly higher delay for the most important type event ($pvalue < 0.01$). Delays under MBTA are roughly equal for all task types.

FMC_TA and SA achieve low delays by sharing most tasks. Cooperative execution also directly results in higher utility through the capability function, but this is only for the tasks of types 1 and

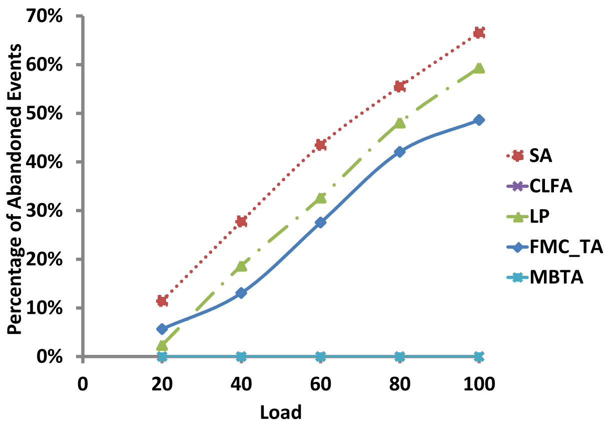

Fig. 9. Average percentage of abandoned events for centralized algorithms.

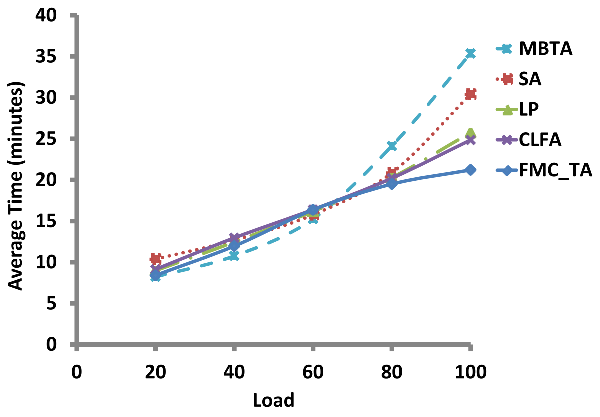

Fig. 10. Average execution delay for centralized algorithms.

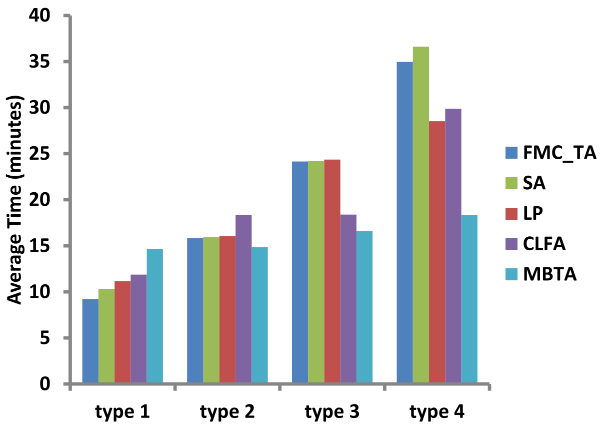

Fig. 11. Average arrival time to events of different types for varying centralized algorithms.

Table 1. Percentage of Shared Tasks

	Load				
Algorithm	20	40	60	80	100
FMC_TA	100	100	99.3	97.4	93.9
SA	99.9	99.7	98.7	94.4	82.8

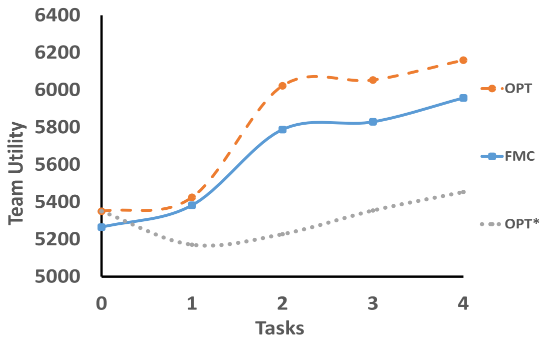

Fig. 12. Team utility as a function of the number of tasks with $max_q(CAP(j, q)) > 1$.

2. Table 1 presents an average percentage of shared tasks for FMC_TA and SA; other algorithms do not allow sharing. FMC_TA shares a greater percentage of tasks. Sharing tasks allows agents to divide the workload, and thus begin working on subsequent tasks more quickly. SA also benefits from this, but makes agents share fewer tasks at high loads than FMC_TA.

In conclusion of this set of comparisons, FMC_TA is based on the Fisher market-clearing solution, which achieves implicitly a unique combination of fairness and optimality properties in its allocation. Other methods, which we compare with, that directly optimize the team utility (like SA) try to optimize the reduced problem that the Fisher market receives as input directly (LP), or specifically designed heuristics (CFLA, MBTA) are not able to generate such an efficient allocation to ad hoc coalitions, as achieved by the Fisher market-clearing solution.

5.4 Comparison to an Optimal Solution

The proposed FMC_TA algorithm is a heuristic and offers no guarantees on the distance of the quality of the solution it computes in comparison to the quality of the optimal solution. Nevertheless, we can provide an indication of this gap by comparing the proposed algorithm to a complete exhaustive search algorithm on small problems where it is feasible to compute the optimal solution. Figure 12 presents a comparison between the team utility of FMC_TA and a complete Branch and Bound, exhaustive search algorithm. The setup is similar to the setup of the experiments presented above, only smaller. All problems included four agents and four tasks. The difference between the problems was in the capability function of the tasks. The tick mark values on the X axis represent the number of tasks, which their maximal capability function value was for $q > 1$, i.e., that the agents derive more utility for sharing them. Each data point in the graph is an average over 100 runs of the algorithm. The results showing that the utility derived by the agents when using FMC_TA is lower than the utility derived from the optimal solution were significant ($p\_value \leq 0.05$). The results were found to be significant with $p \leq 0.05$. The graph also includes the best result that can be achieved without sharing tasks (the opt* version). It is clear that FMC_TA, in which sharing

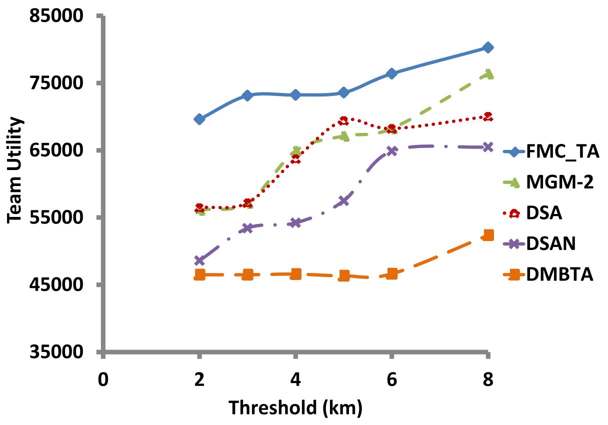

Fig. 13. Team utility for varying threshold distances on shifts with 60 events.

is inherent, is able to find solutions that approximate the optimal solution much better than any method that does not offer sharing.

5.5 Evaluation of Distributed Algorithms

We implemented the distributed version of *FMC_TA* using the algorithm proposed in Reference [42] to solve Fisher market-clearing problem. We compared the results to DMBTA (a distributed version of MBTA) as well as general-purpose algorithms solving a distributed constraint optimization problem (DCOP) formulation of LEP. Unlike the full knowledge centralized system, each agent in the distributed system only knows of tasks in its local environment, i.e., tasks within the threshold distance of the agent's location.

In the DCOP formulation of LEP, each agent a_i holds a sequence of variables $z_{i_1}, \ldots, z_{i_k}$, where k is the maximum number of fractions of tasks an agent can be allocated. The domain of each variable is the set of tasks within the threshold distance and a patrol task. The order of the variables represents the order in which the agent will perform these tasks, i.e., a_i will first perform the fraction of task assigned to variable z_{i_1}, then the fraction of task assigned to z_{i_2}, and so on. Utilities were represented by a constraint between all variables for each task. There are also constraints limiting each agent to one patrol task, which must be scheduled after all events.

Results for three DCOP algorithms are presented: the Distributed Stochastic Algorithm (DSA) [43], Distributed Simulated Annealing (DSAN) [2], and MGM-2 [20]. These algorithms are known to produce high-quality results while (in contrast to algorithms like Max-sum and DALO) their running time does not increase exponentially with the number of agents involved in each constraint. This is extremely important, because in *LEP* constraints generally involve many agents. The comparison algorithms are local-search algorithms that iteratively improve a selected assignment. The agents running each algorithm select their previous allocation as their initial assignment following a dynamic event. Following each dynamic event, the agents performed 400 iterations of the DCOP algorithm before deciding on the allocation.

Figure 13 presents the team utility for threshold distances from 2 km to 8 km for shifts with load of 60. It is clear that the distributed algorithms find high-quality solutions, but *FMC_TA* dominates, achieving higher utility under all thresholds ($pvalue < 0.04$). *FMC_TA*, for thresholds equal or larger than 5 km, finds solutions of statistically equal quality to that of the centralized algorithm. DMBTA does poorly at low and high thresholds.

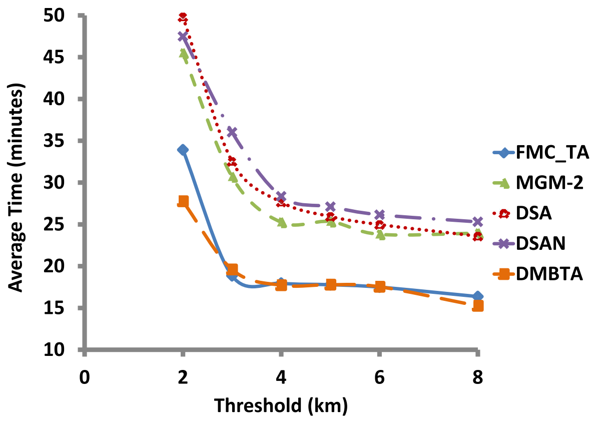

Fig. 14. Average execution delay for varying threshold distances for shifts with 60 events.

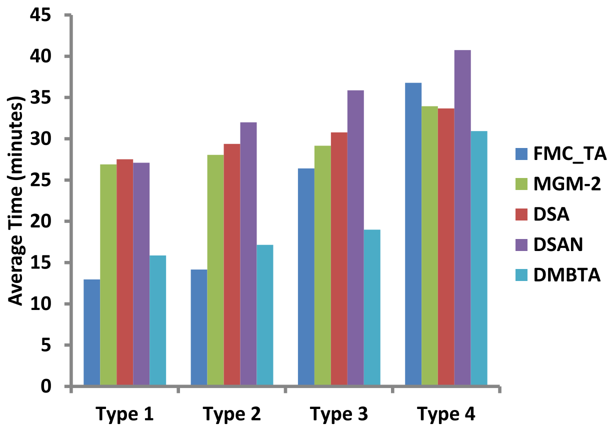

Fig. 15. Average arrival time to events of different types for varying distributed algorithms.

FMC_TA together with DMBTA find solutions with the lowest average task execution delay, as shown in Figure 14. *FMC_TA* was especially successful in prioritizing important tasks to minimize execution delay, while the delays for tasks of all types were roughly equal in schedules found by the other algorithms—see Figure 15, which presents the execution delay as a function of the event type for shifts with a load of 60 and a 3-km threshold.

5.6 Evaluation of Larger Problems

To investigate the resilience of *FMC_TA* to problem scaling, we performed experiments on three types of larger problems. The settings of the first type were similar to the settings in the experiments we described above, except that they included a larger number of agents, either 16, 25, 36, or 49. In the second type of problem, the number of agents remained 9, while the size of the city and the number of tasks grew. The third type of problems included both a larger number of agents and a larger city. The size of the city, for this type of problem, was determined in proportion with the number of agents. The used proportions were: the ratio between the number of agents and city size was $\frac{1}{4}$, ratio between the number of agents and the police patrols was 1, and ratios between

Table 2. Example of Proportions for Enlarged Problems

			Number of tasks				
Agents	City size	Neighborhoods	1	2	3	4	5
9	6×6	9	20	40	60	80	100
16	10×10	16	36	71	107	142	178
25	10×10	25	56	111	167	222	278
36	10×10	36	80	160	240	320	400
49	10×10	49	109	218	327	436	544

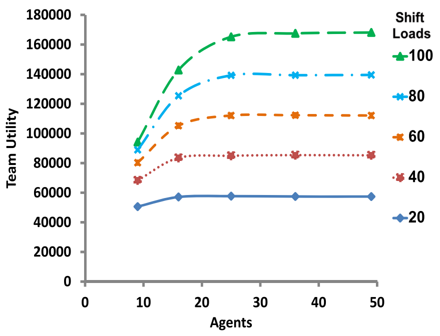

Fig. 16. Accumulated team utility function as a function of the number of agents for varying shift loads.

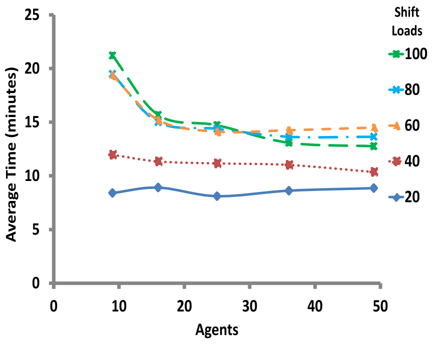

Fig. 17. Average execution delay function as a function of the number of agents for varying shift loads.

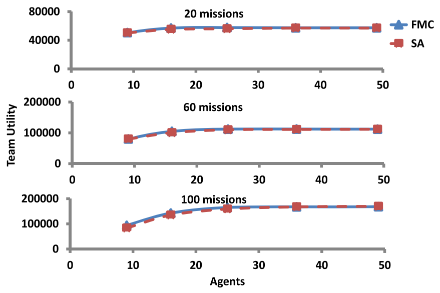

Fig. 18. Accumulated team utility function as a function of number of agents for varying shift loads. FMC_TA vs. SA.

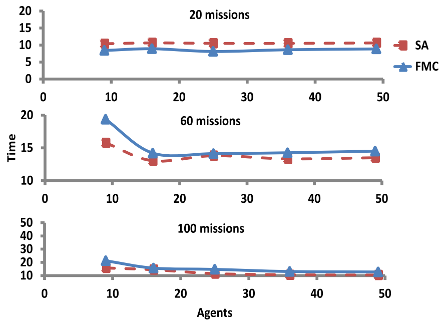

Fig. 19. Average execution delay function as a function of the number of agents for varying shift loads. FMC_TA vs. SA.

the number of agents and the shift load were $\frac{9}{20}$, $\frac{9}{40}$, $\frac{9}{60}$, $\frac{9}{80}$, and $\frac{9}{100}$. An example of the proportions that were examined can be seen in Table 2.

On each type of problem, we compared *FMC_TA* with the best competing state-of-the-art algorithm, Simulated Annealing.

The results of experiments including the first type of problem are presented in Figures 16, 17, 18, and 19. The results indicate that as the number of agents increases (and loads remain the same), up to 25 agents, the agents manage to cope better with the incoming events. In all metrics (SW, execution delay, abandonment rate, and sharing rate) there is significant improvement when the

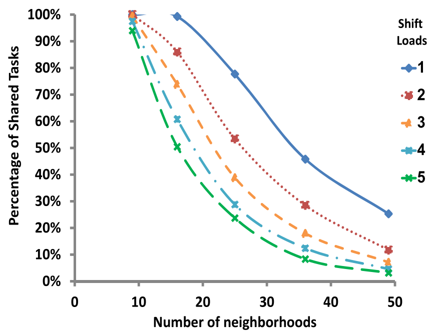

Fig. 20. Average percentage of shared tasks for cities of various sizes.

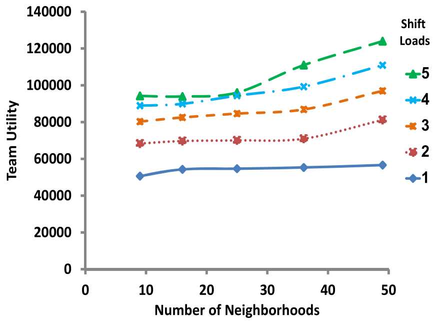

Fig. 21. Accumulated team utility for cities of various sizes.

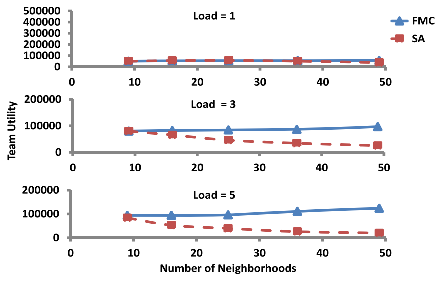

Fig. 22. Accumulated team utility for cities of various sizes. FMC_TA vs. SA

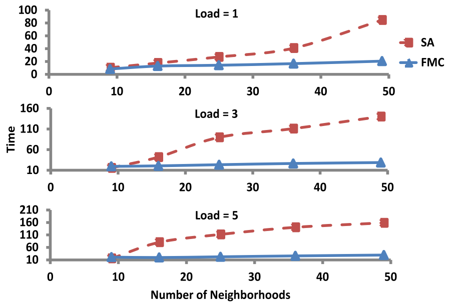

Fig. 23. Average execution delay for cities of various sizes. FMC_TA vs. SA

first agents are added (9, 16, and 25 agents) and a convergence starting at 25 agents. When we compare the *FMC_TA* results to SA, we can see the same effects. Raising only the number of agents improves the performance of *FMC_TA* and SA equally.

The results of the experiments performed on the second type of problems, where we fix the number of agents and increase the other parameters of the problem, i.e., the size of the city (number of neighborhoods) and the number of missions per shift, are presented in Figures 20, 21, 22, and 23.

These results demonstrate that when we increase only the size of the problem and maintain the number of agents, there are more events per agent to cope with at the same time and the agents are inherently forced (by the FMC_TA mechanism) to work separately to cover as many of the incoming events as possible. Thus, the sharing rate declines. Less sharing results in higher time spent on missions in average and growing execution delays for new missions. However, team utility increases. The FMC_TA algorithm prioritizes the important events, hence the agents handle events with high importance first, and the team utility is increased, although other metrics are compromised. The comparison to SA reveals a larger advantage in the favor of FMC_TA for larger problems. Apparently, SA fails to cope with high complexity and a large domain as efficiently as FMC_TA.

Figures 24, 25, 26, and 27 present the results of the experiments including problems of the third type.

They indicate that the quality of the FMC_TA solution has not been affected by higher complexity of the problems when the number of agents grows proportionally. The team utility grows

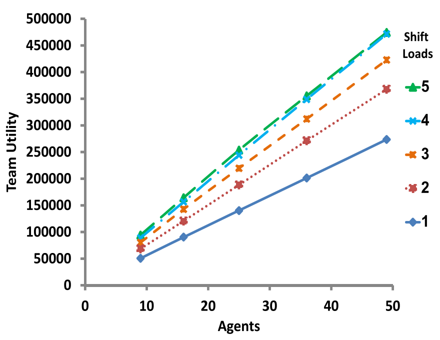

Fig. 24. Accumulated team utility per number of agents for various loads.

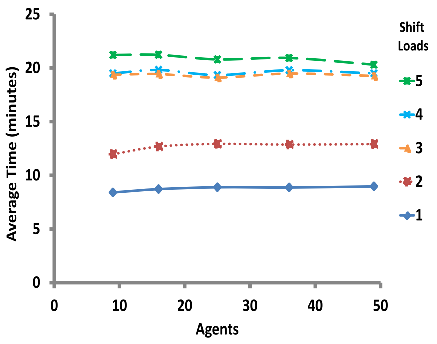

Fig. 25. Average execution delay per number of agents for various loads.

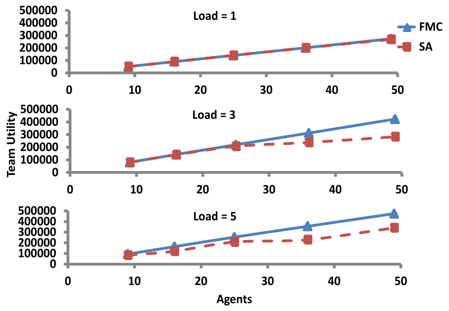

Fig. 26. Accumulated team utility as a function of the number of agents for various shift loads. FMC_TA vs. SA.

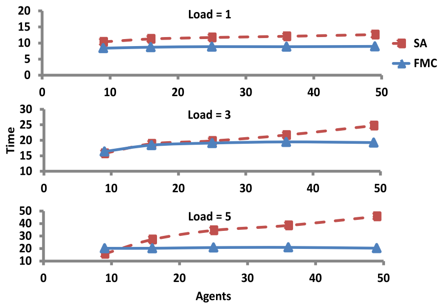

Fig. 27. Average execution delay as a function of the number of agents for various shift loads. FMC_TA vs. SA.

linearly as a function of the size of the city, while other metrics have not been affected and remain the same as in the small problem.

As for SA, the team utility grows as the number of agents grows but not as fast as for FMC_TA. This is consistent across several metrics. The high complexity causes higher execution delay, lower percentage of cooperation, and higher percentage of interrupted events that explicitly affect team utility.

6 CONCLUSION AND FUTURE WORK

Realistic multi-agent team applications often require rapid solving of hard dynamic task allocation problems that include temporal and spatial constraints. Such problems are commonly solved using state-of-the-art incomplete algorithms for solving combinatorial optimization problems, e.g., Simulated annealing, or specifically designed algorithms that aim at maximizing the special optimization criteria of such problems.

In this article, we propose FMC_TA, a novel task allocation algorithm based on the market-clearing paradigm. FMC_TA first finds allocations that are fair (envy-free) and efficient (Pareto optimal) in a simplified version of the problem and then heuristically schedules the allocations, taking into account inter-agent constraints on shared tasks.

Our experimental results show that FMC_TA effectively and efficiently allocates and schedules tasks for agents in the complex, dynamic settings characteristic of law enforcement, which is common in other multi-agent applications. The comparison with LP demonstrates that the two-stage allocate-schedule approach is not solely responsible for the high quality of its solutions. Instead, it is the combination of fairness and efficiency of the allocations in the restricted problem that result in allocations that share important tasks, enabling synergistic cooperation that leads to higher quality task execution while driving down delays.

Despite this success, we note that the use of linear personal utility functions as an input for the Fisher market results in agents being indifferent between performing tasks on their own and sharing them with others. In future work, we intend to investigate whether the use of convex personal utility functions can incentivize more cooperation and improve the results. For the simplicity of our simulation, we assumed a uniform spatial distribution of tasks. An interesting future research question could be whether the spatial distribution of tasks affects the relative performance of task allocation algorithms.

REFERENCES

[1] Sofia Amador, Steven Okamoto, and Roie Zivan. 2014. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1495–1496.

[2] M. Arshad and M. C. Silaghi. 2004. Distributed simulated annealing. In *Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems: Frontiers in Artificial Intelligence and Applications. IOS Press.*

[3] William C. Brainard and Herbert E. Scarf. 2000. *How to Compute Equilibrium Prices in 1891.* Cowles Foundation for Research in Economics - Yale University.

[4] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. 2014. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Auton. Agents Multi-Agent Syst.* 28, 1 (2014), 101–125.

[5] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. Robotics* 25, 4 (2009), 912–926.

[6] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, and Laxmidhar Behera. 2015. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *J. Intell. Robotic Syst.* 80, 1 (2015), 33–58.

[7] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. 2002. Market equilibrium via a primal-dual-type algorithm. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS'02)*. 389–395.

[8] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. 2006. Market-based multirobot coordination: A survey and analysis. *Proc. IEEE* 94, 7 (2006), 1257–1270.

[9] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. 2017. Distributed on-line dynamic task assignment for multi-robot patrolling. *Auton. Robots* 41, 6 (2017), 1321–1345.

[10] Alessandro Farinelli, Alex Rogers, and Nick R. Jennings. 2014. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Auton. Agents Multi-Agent Syst.* 28, 3 (2014), 337–380.

[11] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*. 639–646.

[12] S. Shaheen Fatima and Michael Wooldridge. 2001. Adaptive task resources allocation in multi-agent systems. In *Proceedings of the 5th International Conference on Autonomous Agents*. ACM, 537–544.

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

[14] J. Godoy and M. Gini. 2013. Task allocation for spatially and temporally distributed tasks. In *Intelligent Autonomous Systems 12*. Springer, 603–612.

[15] Yichuan Jiang. 2015. A survey of task allocation and load balancing in distributed systems. *IEEE Trans. Parallel Distrib. Syst.* 27, 2 (2015), 585–599.

[16] E. G. Jones, M. B. Dias, and A. Stentz. 2007. Learning-enhanced market-based task allocation for oversubscribed domains. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*. IEEE, 2308–2313.

[17] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain. 2006. The power of sequential single-item auctions for agent coordination. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21.

[18] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and A. Huberman. 2005. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiag. Grid Syst.* 1, 3 (2005), 169–182.

[19] K. S. Macarthur, R. Stranders, S. Ramchurn, and N. R. Jennings. 2011. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI'11)*. AAAI Press, 701–706.

[20] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. 2004. Distributed algorithms for DCOP: A graphical-game-based approach. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*. 15–17.

[21] M. Nanjanath and M. Gini. 2010. Repeated auctions for robust task execution by a robot team. *Robotics Auton. Syst.* 58, 7 (2010), 900–909.

[22] Ernesto Nunes and Maria Gini. 2015. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

[23] Ernesto Nunes, Marie Manner, Hakim Mitiche, and Maria Gini. 2017. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics Auton. Syst.* 90, C (April 2017), 55–70.

[24] James Parker and Maria Gini. 2014. Tasks with cost growing over time and agent reallocation delays. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 381–388.

[25] James Parker, Ernesto Nunes, Julio Godoy, and Maria L. Gini. 2016. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *J. Field Robotics* 33, 7 (2016), 877–900.

[26] T. O. Paulussen, N. R. Jennings, K. S. Decker, and A. Heinzl. 2003. Distributed patient scheduling in hospitals. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1224–1229.

[27] Graham Pinhey, John A. Doucette, and Robin Cohen. 2018. Using reservations for multiagent resource allocation with costly preemption. *Multiag. Grid Syst.* 14, 3 (2018), 219–242.

[28] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. 2012. Distributed chance-constrained task allocation for autonomous multi-agent teams. In *Proceedings of the American Control Conference (ACC'12)*. 4528–4533.

[29] D. Poole and A. K. Mackworth. 2010. *Artificial Intelligence—Foundations of Computational Agents*. Cambridge University Press.

[30] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. 2010. Decentralized coordination in RoboCup Rescue. *Comput. J.* 53, 9 (2010), 1447–1461.

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

[32] C. R. Reeves. 1993. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc.

[33] J. H. Reijnierse and J. A. M. Potters. 1998. On finding an envy-free Pareto-optimal division. *Math. Progr.* 83 (1998), 291–311.

[34] A. Schoneveld, J. F. de Ronde, and P. M. A. Sloot. 1997. On the complexity of task allocation. *J. Complex.* 3 (1997), 52–60.

[35] O. Shehory and S. Kraus. 1998. Methods for task allocation via agent coalition formation. *Artific. Intell.* 101, 1 (1998), 165–200.

[36] Steven Skiena. 2008. *The Algorithm Design Manual* (2nd ed.). Springer.

[37] W. E. Walsh and M. P. Wellman. 1998. A market protocol for decentralized task allocation. In *Proceedings of the International Conference on Multi-Agent Systems*. 325–332.

[38] Changyun Wei, Koen V. Hindriks, and Catholijn M. Jonker. 2016. Dynamic task allocation for multi-robot search and retrieval tasks. *Appl. Intell.* 45, 2 (2016), 383–401.

[39] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. 2001. Analyzing market-based resource allocation strategies for the computational grid. *Int. J. High Perf. Comput. Appl.* 15, 3 (2001), 258–281.

[40] Dayong Ye, Minjie Zhang, and Athanasios V. Vasilakos. 2016. A survey of self-organization mechanisms in multiagent systems. *IEEE Trans. Syst., Man, and Cyber.: Syst.* 47, 3 (2016), 441–461.

[41] Harel Yedidsion, Roie Zivan, and Alessandro Farinelli. 2018. Applying max-sum to teams of mobile sensing agents. *Eng. Appl. Artific. Intell.* 71 (2018), 87–99.

[42] L. Zhang. 2011. Proportional response dynamics in the Fisher market. *Theor. Comput. Sci.* 412, 24 (2011), 2691–2698.

[43] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. 2005. Distributed stochastic search and distributed breakout: Properties, comparison, and applications to constraints optimization problems in sensor networks. *Artific. Intell.* 161:1-2 (Jan. 2005), 55–88.

[44] R. Zivan, S. Okamoto, and H. Peled. 2014. Explorative anytime local search for distributed constraint optimization. *Artific. Intell.* 211 (2014).

[45] R. Zlot and A. Stentz. 2006. Market-based multirobot coordination for complex tasks. *Int. J. Robotics Res.* 25, 1 (2006), 73–101.

Received January 2019; revised July 2019; accepted August 2019