Market-based Risk Allocation Optimization [*]

Masahiro Ono and Brian C. Williams
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{hiro_ono, williams}@mit.edu

ABSTRACT

This paper proposes *Market-based Iterative Risk Allocation* (*MIRA*), a new market-based decentralized optimization algorithm for multi-agent systems under stochastic uncertainty, with a focus on problems with continuous action and state space. In large coordination problems, from power grid management to multi-vehicle missions, multiple agents act collectively in order to maximize the performance of the system, while satisfying mission constraints. These optimal action plans are particularly susceptible to risk when uncertainty is introduced. We present a decentralized optimization algorithm that minimizes the system cost while ensuring that the probability of violating mission constraints is below a user-specified upper bound.

We build upon the paradigm of *risk allocation* [13], in which the planner optimizes not only the sequence of actions, but also its allocation of risk among state constraints. We extend the concept of risk allocation to multi-agent systems by highlighting risk as a resource that is traded in a computational market. The equilibrium price of risk that balances the supply and demand is found by an iterative price adjustment process called *tâtonnement* (also known as *Walrasian auction*). Our work is distinct from the classical tâtonnement approach in that we use Brent's method to provide fast guaranteed convergence to the equilibrium price. The simulation results demonstrate the efficiency and optimality of the proposed decentralized optimization algorithm.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms

Keywords

Chance constrained optimal planning, Decentralized optimization, Tâtonnement, Walrasian auction, Continuous resource allocation

[*]This research is funded by The Boeing Company grant MIT-BA-GTA-1

1. INTRODUCTION

1.1 Motivation

There is an increasing need for multi-agent systems that perform optimal planning under uncertainty. An example is planning and control of power grid systems [3]. A power grid consists of a number of generators and electric transformers whose control should be carefully planned in order to maximize efficiency. A significant issue in power grid planning is the uncertainty in demand for energy by consumers. As the use of renewable energy, such as solar and wind power, become more popular, uncertainty in supply increases due to weather conditions.

Another example is the Autonomous Ocean Sampling Network (AOSN) [18], which consists of multiple automated underwater vehicles (AUVs), robotic buoys, and aerial vehicles. AOSN should maximize science gain while being exposed to external disturbances, such as tides and currents.

To deal with such problems, we developed *Market-based Iterative Risk Allocation* (MIRA), a multi-agent optimization algorithm that operates within user-specified risk bounds. The scope of this paper is a dynamic system with continuous state and action space under stochastic uncertainty.

1.2 Overview

Optimization of action sequence under uncertainty, and risk allocation.

When planning action sequence under uncertainty, there is always a risk of failure that should be avoided. However, in many cases, performance can be improved only by taking extra risk. We can reach a destination faster by driving at a faster speed and accepting a higher risk of an accident. Hannibal, a Carthaginian military commander in the third century B.C., was able to frustrate the Roman army by taking the great risk of crossing the Alps with 50,000 troops. As seen in these examples, risk and performance are in a trade-off relationship. In other words, risk is a resource that can be spent to improve the performance of the system.

Without taking any risk, nothing can be done; however, no one dares to take unlimited risk. Although the sensitivity for risk varies from person to person, everyone somehow balances risk and performance to find the optimal action sequence.

There are three main ways to formulate the trade-off problem of risk and performance; the first is to set a negative utility for failure (i.e. penalty), and maximize the expected total utility (the utilitarian approach, such as MDP [2][11][12]); the second is to set upper bound on risk and maximize performance within this bound [13][16]; the third is to set lower bound on performance and minimize risk. It is up to the user to choose which formulation to use

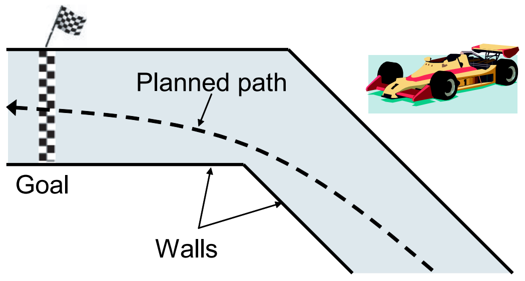


Figure 1: Risk allocation in a race car path planning scenario. A large portion of risk is allocated to the corner, since taking a risk (approaching the wall) at corner results in greater time saving than taking the same risk along straightaway.

according to her needs and requirements.

Our focus is on the second approach: performance maximization with an upper-bound on risk. An example problem is to drive a car as fast as possible while limiting the probability of a crash to 0.01%. This formulation is particularly useful for optimal planning and control problems that involve high-impact low-probability risk such as loss of life.

With this formulation, [13] showed that we should optimize not only the sequence of actions but also the *risk allocation* in order to maximize the performance under a risk bound .

The example shown in Figure 1 illustrates the concept of risk allocation. A race car driver wants to plan a path to get to the goal as fast as possible. However, crashing into the wall leads to a fatal accident, so he wants to limit the probability of a crash to 0.01%. An intelligent driver would plan a path as shown in Figure 1, which runs mostly in the middle of the straightaway, but gets close to the wall at the corner. This is because taking a risk (i.e. approaching the wall) at the corner results in a greater time saving than taking the same risk along the straightaway; in other words, the utility of taking risk is greater at the corner than the straightaway. Therefore the optimal path plan allocates a large portion of risk to the corner, while allocating little to the straightaway. As illustrated by this example, *risk allocation* needs to be optimized across the constraints, in order to maximize the performance.

The optimal controller then needs to generate an optimal action sequence that abides to the allocated risk at each constraint.

Risk allocation for multi-agent system.

Past work on risk allocation [6][13][14] focused on single agent problems.

In this work we extend the concept of risk allocation to multi-agent systems. Figure 2 shows an example of a multi-agent system with two unmanned air vehicles (UAVs), whose mission is to extinguish a forest fire. A water tanker drops water while a reconnaissance vehicle monitors the fire with its sensors. The loss of either vehicle results in a failure of the mission. Two vehicles are required to extinguish the fire as efficiently as possible, while limiting the probability of mission failure to a given risk bound, say, 0.1%. The water tanker can improve efficiency by flying at a lower altitude, but it involves risk. The reconnaissance vehicle can also improve the data resolution by flying low, but the improvement of efficiency is not as great as the water tanker. In such a case a plausible plan is to allow the water tanker to take a large portion of risk by flying low, while keeping the reconnaissance vehicle at a high altitude to avoid risk. This is because the utility of taking risk (i.e. flying low)

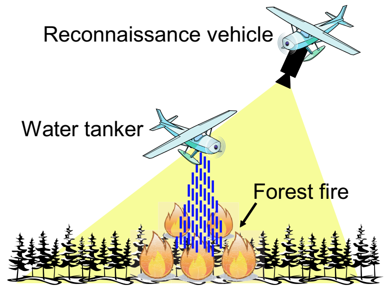


Figure 2: Risk allocation for multi-UAV fire-fighting system. The water tanker is allowed to fly low since it is allocated larger risk than the reconnaissance vehicle.

is greater for the water tanker than for the reconnaissance vehicle.

The optimal risk allocation for multi-agent systems can be found by applying the same algorithm as the single agent problems, such as [6][13][16], with extended state variable that include all agents. However, this approach requires centralized computation, which has an issue of scalability.

In this paper we propose a novel *decentralized* algorithm that finds the globally optimal risk allocation among multiple agents.

Market-based risk allocation using tâtonnement.

Our approach is to use the market-based mechanism. In a computational market each agent demands risk in order to improve its own performance. However, it cannot take risk for free; it has to purchase it from the market at a given price.

Agents are price takers. Given the price, each agent computes the optimal amount of risk to take (i.e., *demand for risk*) by solving the optimization problem where the objective function is the summation of the original cost function and the payment for the risk. The optimal action sequence and the internal risk allocation are also determined by solving the optimization problem, just as in the single-agent case described before. The demand from each agent can be seen as a function of the price of risk (*demand curve*). Typically, the higher the price is, the less each agent demands. Each agent has a different demand curve according to its sensitivity to risk. On the other hand, the supplier of the risk is the user. She supplies the fixed amount of risk by specifying the upper-bound of risk the system can take.

The price must be adjusted so that the total demand (*aggregate demand*) becomes equal to the supply. The equilibrium price is found by an iterative process called *tâtonnement* or *Walrasian auction* [17] as follows:

- Increase the price if aggregate demand exceeds supply,
- Decrease the price if supply exceeds aggregate demand,
- Repeat until supply and demand are balanced.

In classical tâtonnement, the price increment is obtained by simply multiplying the excess aggregate demand by a constant. However, the upperbound of the constant that guarantees the convergence is specific to a problem, and is hard to find. Slow convergence speed is also an issue. Our method obtains the price increment in each iteration by computing one step of Brent's method, which is a commonly-used root-finding algorithm with fast and guaranteed convergence [1].

Figure 3 gives the graphical interpretation of the market-based risk allocation in a system with two agents. Agent 1 and Agent 2 have different demand curves, since their utility of taking the same

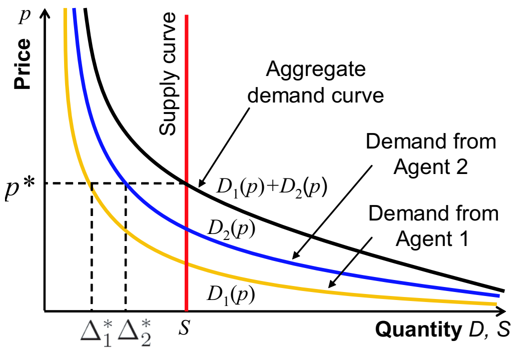


Figure 3: Market-based risk allocation in a system with two agents. Note that we followed the economics convention of placing the price on the vertical axis. The equilibrium price is $p^\star$, and the optimal risk allocation is $\Delta_1^\star = D_1(p^\star)$ for Agent 1 and $\Delta_2^\star = D_2(p^\star)$ for Agent 2.

risk is different. The aggregate demand curve is obtained by adding the two demand curves horizontally. The supply curve is a vertical line since it is constant. The equilibrium price $p^\star$ lies at the intersection of the aggregate demand curve and the supply curve. The optimal risk allocation for the two agents corresponds to their demands at the equilibrium price ($\Delta_1^\star$ and $\Delta_2^\star$ in Figure 3).

It is proven in a later section that the performance of the entire system is maximized at the equilibrium price, although each agent only maximizes its own utility. The only information that each agent needs to communicate in each iteration is price and demand, both of which are a scalar value. These are desirable features for distributed systems.

MIRA - Decentralized optimization of risk allocation.

Our proposed algorithm, MIRA (Market-based Iterative Risk Allocation), optimizes risk allocation between agents, internal risk allocation of each agent, and action sequences of each agent concurrently.

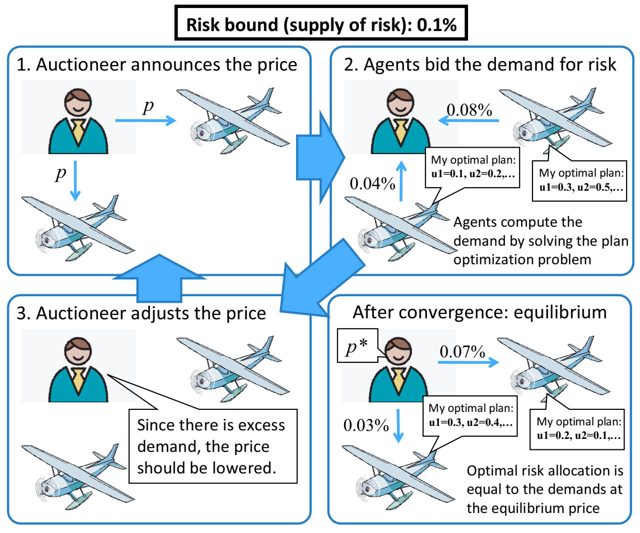
Figure 4 illustrates the Market-based Iterative Risk Allocation (MIRA) algorithm. The tâtonnement process is repeated until it converges to the equilibrium price. Risk is not allocated until the algorithm converges. The optimal action sequence and the internal risk allocation are also obtained as the by-product of the demand optimization problem (Step 2 in the Figure 4).

Figure 5 shows how MIRA algorithm breaks down the risk for individual constrains in each agent. The risk bound for the system is given by the user (A). Risk is optimally allocated to agents through tâtonnement (B). Each agent optimizes internal risk allocation when computing the demand in each iteration of MIRA (C). At the same time, the action sequence is optimized according to the internal risk allocation (D).

1.3 Related Work

MDP-based algorithms have been mainly used to solve multi-agent planning problems under uncertainty in discrete domains [2] [11][12]. M-DPFP algorithm proposed by [9] can solve problems with continuous resources. Our problem formulation is different from MDP-based approaches in that we set an upper bound on risk (*chance constraint*) and maximize performance within this bound, instead of maximizing utility. Also, our focus is on the problem with continuous state and action space.

Optimal control under uncertainty with chance constraint is in-



Risk bound (supply of risk): 0.1%
1. Auctioneer announces the price
p
p
2. Agents bid the demand for risk
0.08%
0.04%
My optimal plan: u1=0.1, u2=0.2,...
My optimal plan: u1=0.3, u2=0.5,...
Agents compute the demand by solving the plan optimization problem
3. Auctioneer adjusts the price
Since there is excess demand, the price should be lowered.
After convergence: equilibrium
p^*
0.07%
0.03%
My optimal plan: u1=0.3, u2=0.4,...
My optimal plan: u1=0.2, u2=0.1,...
Optimal risk allocation is equal to the demands at the equilibrium price

Figure 4: Illustration of MIRA algorithm. Risk is allocated to agents through tâtonnement; their continuous action sequences are also optimized in the loop when computing the demand at the given price.

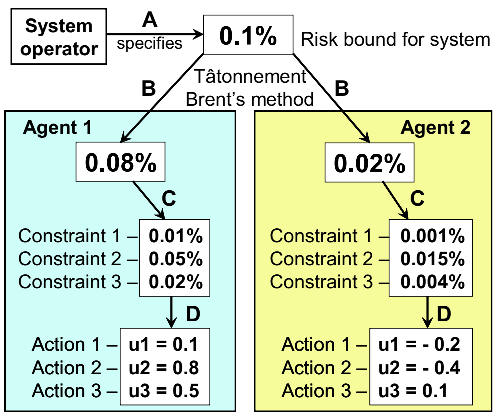


Figure 5: Distribution of risk in MIRA algorithm.

tensively researched in the robust model predictive control (RMPC) community. Due to the difficulty of handling the chance constraint analytically, past work used either a very conservative bound that resulted in large suboptimality [5][19], or a sample-based method [4] that is computationally inefficient. Based on the pioneer work by [16] that proposed the conservative approximation of chance constraint by decomposing it into multiple atomic chance constraints, [13] introduced the concept of risk allocation, and developed Iterative Risk Allocation (IRA) algorithm that can optimize risk allocation efficiently, with substantially smaller suboptimality than the past work [14].

Market-based approach has recently been recognized as an effective tool for distributed multi-agent systems in AI community. Although tâtonnement has drawn less attention than auctions, it has been successfully applied to various problems such as the distribution of heating energy in an office building[20], control of electrical power flow[7], and resource allocation in communication networks[8]. In economics, a simple linear price update rule has long been the main subject of study, but the convergence of price can only be guaranteed under a quite restrictive condition[17]. In order to substitute the linear price update rule, various root-finding methods have been employed in agent-based resource allocation algorithms, such as the bisection method[21], Newton method[22], and Broyden's method[20]. However, in general, it is difficult to guarantee quick and stable convergence to the equilibrium price. We employ Brent's method [1] to provide guaranteed convergence with a superlinear rate of convergence by exploiting the fact that a risk, which is treated as a resource in our problem formulation, is a scalar value.

2. RISK ALLOCATION FOR SINGLE-AGENT SYSTEMS

We will first briefly review the mathematical formulation of risk allocation. Our focus on this paper is a problem with continuous state space, although the concept of risk allocation can be used for discrete/hybrid systems [13].

2.1 Formulation

Optimization of action sequence under uncertainty.

We formulate an optimization problem with a chance constraint as follows:

$$\min_{\boldsymbol{u}_{1:T}} \quad J(\boldsymbol{u}_{1:T}) \tag{1}$$

$$\text{s.t.} \quad \forall_t \quad \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{w}_t \tag{2}$$

$$\forall_t \quad \boldsymbol{u}_{\min} \leq \boldsymbol{u}_t \leq \boldsymbol{u}_{\max} \tag{3}$$

$$\Pr\left[\bigwedge_{t=0}^{T}\bigwedge_{n=1}^{N_t} g_{t,n}(\boldsymbol{x}_t) \leq 0\right] \geq 1-\Delta \tag{4}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ are the state vector and action (control input) vector at the tth time step, respectively. The disturbance $\boldsymbol{w}_t$ and the initial state estimation $\boldsymbol{x}_0$ have Gaussian distributions with known mean and variance. Although we focus on Gaussian-distributed uncertainty in this paper for simplicity, our algorithm can be applied to any stochastic uncertainties with quasi-concave distribution. We assume that $J(\cdot)$ and $g_{t,n}(\cdot)$ are convex functions.

A discrete-time stochastic dynamics model of the system is given as (2)(3), and state constraints are imposed as (4). Since violation of any state constraint at any time step is regarded as a mission failure, the probability of satisfying all constraints at all time steps must be more than $1-\Delta$, where Δ is the upper bound of the probability of failure (risk bound). Given a risk bound Δ, the action sequence $\boldsymbol{u}_{1:T} := [\boldsymbol{u}_1 \cdots \boldsymbol{u}_T]^T$ that minimizes the cost J in (1) is obtained as an output by solving the optimization problem. In other words, the user can adjust the risk averseness of the system by specifying the risk bound Δ.

Decomposition of chance constraint.

The chance constraint (4) is hard to evaluate since it involves a probability defined on a multi-dimensional distribution. We decompose this constraint into multiple atomic chance constraints that only involve a single-dimensional distribution, using the following Boole's inequality:

$$\Pr\left[\bigcup_i A_i\right] \leq \sum_i \Pr\left[A_i\right] \tag{5}$$

Observe that, using Boole's inequality (5), following condition (6), together with (7), implies the original chance constraint (4).

$$\forall_{(t,n)} \quad \Pr\left[g_{t,n}(\boldsymbol{x}_t) \leq 0\right] \geq 1-\delta_t^n \tag{6}$$

$$\sum_{t=1}^{T}\sum_{n=1}^{N_t} \delta_t^n \leq \Delta \tag{7}$$

$$\forall_{(t,n)} \quad \delta_t^n \geq 0 \tag{8}$$

Therefore, the original chance constraint (4) can be replaced with (6) and (7). Since we introduced new variables δ_t^n, the cost J in (1) needs to be optimized over δ_t^n as well as the sequence of actions $\boldsymbol{u}_{1:T}$:

$$\min_{\boldsymbol{\delta},\boldsymbol{u}_{1:T}} \quad J(\boldsymbol{u}_{1:T}) \tag{9}$$

where $\boldsymbol{\delta} = \left[\delta_0^1 \; \delta_0^2 \; \cdots \delta_T^{N_T-1} \; \delta_T^{N_T}\right]^T$.

We now have the revised constrained optimization problem defined by (9) with constraints (2)(3)(6)(7).

2.2 Risk allocation

The newly introduced variable $\boldsymbol{\delta}$ is the mathematical representation of *risk allocation*. In (6), each single constraint at each time step has its own risk bound δ_t^n; in other words, δ_t^n is the amount of risk allocated to the nth constraint at the tth time step. Eq.(7) states that the summation of all individual risk bound is upper-bounded by the original risk bound Δ; therefore risk is regarded as a resource with total amount Δ. In order to obtain the maximum performance (minimum cost), the risk allocation needs to be optimized (9), just as resource allocation problems.

The risk allocation optimization problem can be solved efficiently by a two-stage algorithm called Iterative Risk Allocation (IRA) [14]. Alternatively it can also be solved by a single shot optimization [6]. We employ the latter approach in this work.

3. DECENTRALIZED RISK ALLOCATION FOR MULTI-AGENT SYSTEMS

We first formulate the optimization problem under uncertainty for multi-agent systems in a centralized manner, and then derive the decentralized formulation using Karush-Kuhn-Tucker (KKT) conditions of optimality. We will then observe that economic concepts such as price, demand, and supply appear in the resulting formulation.

3.1 Formulation

Optimization of action sequence under risk for multi-agent system.

In a multi-agent system such as the UAV fire fighting system illustrated in Figure 2, the failure of one agent leads to a failure of the entire system. In a manned system, loss of one crew member is regarded as a failure of the mission. Therefore the user wants to set an upper-bound on the probability of having at least one agent fail.

With the same discussion as in the previous section, the following bound is obtained by using Boole's inequality (5):

$$\sum_{i=1}^{I} \Delta_i \leq S \tag{10}$$

where Δ_i is the upper bound on the probability of failure of the ith agent, I is the number of agents in the system, and S is the upper bound on the probability of failure of the entire system (i.e. the total amount of risk the entire system is allowed to take). Note that Δ was a given constant in the single-agent case (4)(7), but now each Δ_i is a decision variable, while S is the new given constant, which is specified by the user.

The performance (cost) of the entire system is defined as the sum of the performance (cost) of all individual agents:

$$J_{sys} = \sum_i J_i(\boldsymbol{u}_{i,1:T}) \tag{11}$$

The optimal control problem under risk for multi-agent systems is formulated as a constrained optimization problem of minimizing (11), subject to the constraints (2)(3)(6)(7) for each agent, and (10).

To simplify this formulation, we define a function[1] $J_i^\star(\Delta_i)$, which is equal to the minimized cost for the ith agent obtained by solving the constrained optimization problem for a single agent (9)(2)(3)(6)(7) given Δ_i:

$$J_i^\star(\Delta_i) = J(\boldsymbol{u}_{i,1:T}^\star)$$

where $\boldsymbol{u}_{1:T}^\star$ is the solution to the single agent optimization problem given Δ_i. An important fact is that function $J_i^\star(\Delta_i)$ is a convex function. See Appendix of [15] for the proof.

Using $J_i^\star(\Delta_i)$, the optimization problem can be rewritten in a simple form as follows:

$$\min_{\Delta_{1:I}} \quad \sum_{i=1}^{I} J_i^\star(\Delta_i) \tag{12}$$

$$\text{s.t.} \quad \sum_i \Delta_i \leq S \tag{13}$$

This formulation describes a centralized approach where the action sequences and risk allocations of all agents are planned in a single optimization problem. We will next derive the decentralized formulation using the KKT conditions of optimality.

Decentralized optimization.

We build upon the resource allocation algorithm proposed by [20], with an modification of using the KKT conditions for optimality instead of the method of Lagrange multipliers, since risk (resource) is bounded by inequality (13) in our problem formulation.

The KKT conditions of the optimization problem (12)(13) are: [2]

$$\left.\frac{dJ_i^\star}{d\Delta_i}\right|_{\Delta_i^\star} + p = 0 \tag{14}$$

$$\sum_i \Delta_i^\star \leq S \tag{13}$$

$$p \geq 0 \tag{15}$$

$$p\left(\sum_i \Delta_i^\star - S\right) = 0 \tag{16}$$

where p is the Lagrange multiplier corresponding to the constraint (13). This is the necessary and sufficient condition for optimality, since $J_i^\star(\Delta_i)$ is convex.

[1]It is often not possible, and not necessary as well, to obtain the function $J_i^\star(\Delta_i)$ in a closed form; in practice $J_i^\star(\Delta_i)$ is evaluated simply by solving the optimization problem (9)(2)(3)(6)(7), with an extra term $p\Delta_i$ added to the objective function (9).

[2]We assume the differentiability of $J_i^\star(\Delta_i)$ here; in fact, since $J_i^\star(\Delta_i)$ is a convex function, it is continuous and differentiable at all but at most countably many points in its domain; we can obtain the same result for the points where it is not differentiable by using extended KKT condition with subgradient.

Observe that (14) is also the optimality condition for the following unconstrained optimization problem:

$$\min_{\Delta_i} J_i^\star(\Delta_i) + p\Delta_i \tag{17}$$

Therefore solving the optimization problem (12) and (13) is equivalent to solving I independent optimization problems (17) with common parameter p, which is determined by (13), (15), and (16). Since (17) contains only the variables related to the ith agent, it can be solved by each agent in a decentralized manner. Each agent optimizes its internal risk allocation $\boldsymbol{\delta}$ (Figure 5-C) and action sequence $\boldsymbol{u}_{1:T}$ (Figure 5-D) by solving (17).

3.2 Economic Interpretation

The economic interpretation of these mathematical manipulations becomes clear by regarding the Lagrange multiplier p as the *price of risk*. Each agent can reduce the cost (i.e. improve the performance) by taking more risk Δ_i, but not for free. Note that a new term $p\Delta_i$ is added to the cost function (17). This is what the agent has to pay to take the amount of risk Δ_i. The agent must find the optimal amount of risk $D_i(p)$ to minimize the cost plus payment, by solving the optimization problem (17) with a given price p:

$$D_i(p) = \arg\min_{\Delta_i} J_i^\star(\Delta_i) + p\Delta_i. \tag{18}$$

In other words, $D_i(p)$ is the amount of risk the ith agent wants to take at the given price of risk p. Therefore $D_i(p)$ can be interpreted as the ith agent's *demand for risk*. On the other hand, the total amount of risk S can be interpreted as the *supply of risk*.

The optimal price $p^\star$ must satisfy the KKT conditions (13), (15), and (16), with the optimal demands at the price $\Delta_i^\star = D_i(p^\star)$. Such a price $p^\star$ is called the *equilibrium price*.

The condition (16) illustrates the relation between the equilibrium price $p^\star$, optimal demand $\Delta_i^\star$, and supply S; in the usual case where the equilibrium price is positive $p^\star > 0$, the aggregate demand $\sum_i \Delta_i^\star$ must be equal to the supply S; in a special case where the supply always exceeds the demand for all $p \geq 0$, the optimal price is zero $p^\star = 0$. If the aggregate demand always exceeds the supply for all $p \geq 0$, there is no solution that satisfies the primal feasibility condition (13), and hence the problem is infeasible. See Figure 3 for the graphical interpretation.

3.3 Global Optimality

The optimal risk allocation to each agent is equal to its demand for risk at the equilibrium price. This is the globally optimal solution since all the KKT conditions for optimality (13)-(16) are satisfied at the equilibrium price. If the supply always exceeds the demand for all $p \geq 0$, the demand at $p = 0$ is the globally optimal risk allocation.

Therefore, we must find the equilibrium price $p^\star$ that satisfies (13)(15)(16) in order to solve the optimization problem. The next section discusses how MIRA finds such an equilibrium price.

4. TÂTONNEMENT: PRICE ADJUSTMENT MECHANISM

We employ an iterative process called tâtonnement to find the equilibrium price: initialize the price with arbitrary value, and update it in each iteration according to the excess demand (supply) until the demand and supply are balanced (Figure 4).

In the real world economy the demand for a good is typically a monotonically decreasing function of price (people want more if price is less). This is also the case in our computational economy.

By differentiating (14),

$$\frac{dp}{dD_i} = -\left.\frac{d^2 J_i^\star}{d\Delta_i^2}\right|_{\Delta_i = D_i} \leq 0 \tag{19}$$

The inequality comes from the fact that $J_i^\star(\Delta_i)$ is a convex function (See the Appendix of [15] for proof). Since demand is monotonically decreasing, the equilibrium price found by tâtonnement is the sole and globally optimal equilibrium.

The price is updated in each iteration (Step 3 of Figure 4). The price update rule must be carefully chosen for quick and stable convergence to the equilibrium. In the following subsections we investigate two update rules: linear increment and Brent's method [1].

4.1 Linear Price Increment

The following simple price update rule is most extensively explored in the economics literatures:

$$p_{k+1} = \max\left\{p_k + c\left(\sum_i D_i(p_k) - S\right), 0\right\} \tag{20}$$

where p_k is the price in the kth iteration.

With this update rule, the price is guaranteed to converge for *sufficiently small* $c > 0$, if a condition called gross substitutability is satisfied [17]. Although this update rule is commonly studied, it has three issues when computing the equilibrium. First, the condition for convergence (gross substitutability) is very restrictive. Second, the constant parameter c is specific to a problem, and it is very hard to obtain the upper bound for which the convergence is guaranteed. Third, the convergence is slow. In our case, where there is a single kind of resource (risk) exchanged in the market, gross substitutability[3] is implied by the decreasing monotonicity of the demand function. However, the other two issues still exist in our case. We solve these issues by applying Brent's method.

4.2 Brent's method

Mathematically, tâtonnement can be seen as a process of finding a root of the excess demand function: $\sum_i D_i(p) - S$.

Brent's method is a root-finding algorithm that achieves quick and guaranteed convergence, by combining three methods: the bisection method, the secant method, and inverse quadratic interpolation [1]. Another important feature of Brent's method is that it does not require the derivative of $D_i(p)$, which is very hard to obtain.

As far as we know, there is only one past work [23] that applies Brent's method to find the equilibrium price, but in economics literature; no past research has applied Brent's method to resource allocation problems.

This is probably because Brent's method can only take a scalar variable, while resource allocation algorithms typically deal with multiple resources (i.e. vector). However, in a *risk* allocation problem, risk is always a scalar value. Therefore Brent's method can efficiently and reliably find the equilibrium price of risk. It is possible to extend our algorithm, MIRA, to solve multi-resource allocation problems by using a generalization of Brent's method[10] or by decomposing the market so that each market deals with only one kind of resource[21]. However, such extensions of MIRA are out of the scope of this paper.

4.3 MIRA algorithm

Algorithm 1 shows the entire flow of the MIRA algorithm. MIRA has a distributed part and centralized part. The computation of demand (Line 5) is distributed to each agent; by solving (18), each agent obtains the optimal demand at the price given by Line 4, as well as its optimal sequence of actions ($\boldsymbol{u}_{i,1:T}$). The computation of price (Line 7) is centralized; the auctioneer collects the demands from all agents (Line 6), and updates the price using Brent's method according to the excess demand/supply. One of the agents plays the role of the auctioneer.

The computation time of the centralized part is substantially shorter than the distributed part. For example, in the fire-fighter UAV scenario with two agents (see Section 5.1), the computation time of the distributed part is 13.8 seconds while the centralized part takes only 0.046 seconds. Moreover, the number of agents does not influence the computation time of centralized part much, since Brent's method only takes the *aggregate* demand (i.e. the summation of the demands of all agents). Therefore, the centralized part of MIRA does not harm the scalability of the algorithm.

The communication requirements between agents are small; in each iteration, each agent receives a price (Line 4) and transmits its demand (Line 6), both of which are scalars.

The centralized part of the algorithm can be distributed by making all individual agents conduct the same computation of price update simultaneously. In such case the demands of all agents must be shared with all agents, while price needs to be shared only at the first iteration. Although we can remove the centralized auctioneer in this way, there is no advantage in terms of computation time.

5. SIMULATION

We implemented MIRA in Matlab. Non-linear optimization solver SNOPT is used to compute the demand $D_i(p)$, and the Matlab implementation of Brent's method (fzero) is used to find the equilibrium price $p^\star$. Simulations were conducted on a machine with Intel(R) Core(TM) i7 CPU clocked at 2.67 GHz and 8GB RAM.

5.1 Validity

We tested MIRA on the multi-UAV altitude planning problem for the fire-fighting scenario (Figure 2). Figure 6 shows the simulation result. Two vehicles fly at the constant horizontal speed, starting from $d = 0$ at altitude 0.5. The mission is to extinguish the fire at $d = 6, 7$. Both vehicles minimize the flight altitude above the fire, although the water tanker is given 100 times more penalty (cost) of flying at high altitude than the reconnaissance vehicle. Both have uncertainty in altitude, so flying at lower altitude involves more risk. The total risk must be less than 0.1%.

The optimal plan allocates 99.2% of the total risk to the water tanker, while only 0.8% to the reconnaissance vehicle. This is because the utility of taking risk (i.e. flying low) is larger for the water tanker than for the reconnaissance vehicle. As a result, the water tanker flies at a lower altitude.

Both vehicles optimize the internal risk allocation as well. For

Algorithm 1 Market-based Iterative Risk Allocation

```
1: Fix $S$;        //Total supply of risk
2: Initialize $p$;        //Price of risk
3: while $|\sum_i D_i(p) - S| > \epsilon$ and $p > 0$ do
4:     Auctioneer announces $p$;
5:     Each agent computes its demand for risk $D_i(p)$ by solving
       (18);
6:     Each agent submits its demand to the auctioneer;
7:     The auctioneer updates $p$ by computing one step of Brent's
       method;
8: end while
```

[3]In the general equilibrium theory, money is also treated as a goods; therefore, in our case, the gross substitutability is defined as the substitutability between risk and money.

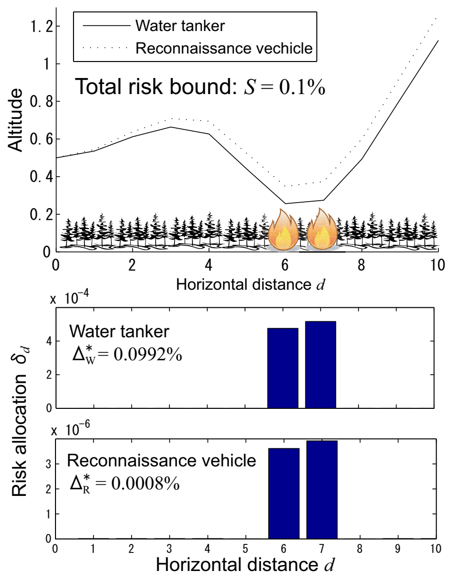


Figure 6: Simulation result of flight altitude planning problem for multi-UAV fire-fighting scenario (see Figure 2). Upper graph: Optimized altitude profile; Lower graphs: Internal risk allocation of each vehicle.

Table 1: Comparison of the computation time of three optimization algorithms. Values are the average of 10 runs with randomly generated constraints.

Number of agents	Computation time [sec]		
	Centralized	Decentralized (linear update)	MIRA
2	13.9	80.6	6.4
4	63.8	540.5	18.1
8	318.5	797.8	37.5

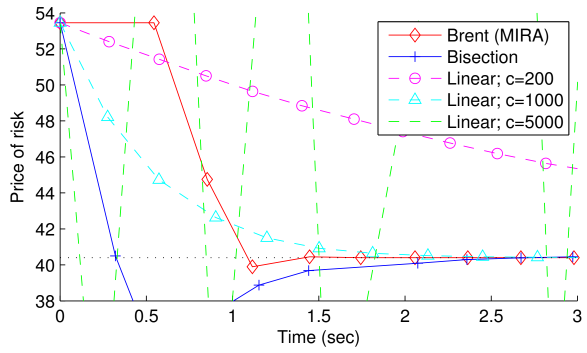


Figure 7: Convergence of different price update methods for tâtonnement

example, the water tanker takes 99.9% of the allocated risk above the fire, at $d = 6$ and 7 (the middle graph in Figure 6).

The optimal action sequence is planned according to this risk allocation; both vehicles dive before the fire, and climb as fast as possible, after they pass the fire (the top graph in Figure 6). This is because there is no benefit of conducting risky low-altitude flight before and after the fire.

These results conform with intuition. The optimality of MIRA is validated by the result that the difference in the optimized cost between MIRA and the centralized algorithm is less than 0.01%, which is accounted by numerical error.

5.2 Efficiency

In order to evaluate the efficiency of the MIRA algorithm, the computation times of the following three algorithms are compared: 1) centralized optimization, 2) decentralized optimization with the linear price update rule (classical tâtonnement), and 3) the proposed algorithm, MIRA, which is a decentralized optimization with Brent's method.

Table 1 shows the result. The three algorithms are tested with different problem sizes - two, four, and eight agents. Each algorithm is run 10 times for each problem size with randomly generated constraints. The parameter c is set so that the price converges in most problems. The average running time is shown in the table. The computation of the distributed algorithms was conducted parallelly. Communication delay is not included in the result.

The computation time of the centralized algorithm quickly grows as the problem size increases. Decentralized optimization with a linear price increment is even slower than the centralized algorithm, although the growth rate of computation time is slower.

MIRA, the proposed algorithm, outperforms the other two for all problem sizes. The advantage of MIRA becomes clearer as the problem size increases.

A counterintuitive phenomenon observed in the result is that the decentralized algorithms (MIRA and decentralized optimization with linear increment) also slow down for large problems, although not as significantly as the centralized algorithm. This is mainly because the iterations of tâtonnement must be synchronized among all agents. When each agent computes its demand for risk by solving the non-linear optimization problem, the computation time diverges from agent to agent, and from situation to situation. In each iteration of tâtonnement, all agents must wait until the slowest agent finishes computing its demand. As a result, tâtonnement process slows down for large problems, as the expected computation time of the slowest agent grows.

5.3 Convergence

Figure 7 shows the convergence of different price update algorithms for tâtonnement on a problem with four agents. Three algorithms are compared: Brent's method, which is employed by our proposed algorithm MIRA, the bisection method, which is used by WALRAS [21], and the linear price update. The linear price update is tested with three different settings of the parameter c in (20).

It is empirically shown from the result that increasing c makes the linear price update method faster, but it diverges when c is too large. The upperbound of c that guarantees the convergence is specific to a problem, and hard to obtain. Brent's method achieves the fastest converges among the three methods. Its convergence is guaranteed without parameter tunings.

5.4 Used Parameters

The horizontal speed of the vehicles is 1 per time step. Hence, $d = t$. The planning window is $1 \leq t \leq 10$. Other parameters are set as follows:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}, \boldsymbol{x}_0 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix},$$

$$u_{\min} = -0.2, u_{\max} = 0.2, \quad g_t(\boldsymbol{x}_t) = -[1\ \ 0]\,\boldsymbol{x}_t + l_t$$

$\boldsymbol{w}_t$ is sampled from zero-mean Gaussian distribution with variance

$$\Sigma_w = \begin{bmatrix} 0.001 & 0 \\ 0 & 0 \end{bmatrix}.$$

l_t is the ground level at t. It is set at zero in the fire-fighter UAV scenario, and randomly generated for the evaluation of computation time. The cost functions are

$$\begin{aligned} J_W &= E\left[[100\ \ 0]\,(\boldsymbol{x}_{6,W} + \boldsymbol{x}_{7,W})\right] \\ J_R &= E\left[[1\ \ 0]\,(\boldsymbol{x}_{6,R} + \boldsymbol{x}_{7,R})\right] \end{aligned}$$

in Section 5.1 (subscript W and R indicate the water tanker and the reconnaissance vehicle respectively), and

$$J_i = E\left[\begin{bmatrix} 1 & 0 \end{bmatrix} (\sum_{t=1}^{10} \boldsymbol{x}_{t,i})\right]$$

in Section 5.2 and 5.3. Note that the expectation of $\boldsymbol{x}_t$ is a function of $\boldsymbol{u}_{1:t-1}$. Therefore J is a function of $\boldsymbol{u}_{1:T}$.

6. CONCLUSION

We have developed Market-based Iterative Risk Allocation (MIRA), a multi-agent optimization algorithm that operates within user-specified risk bounds. It was built upon the concept of risk allocation. The key innovations presented in the paper include 1) extension of the concept of risk allocation to multi-agent system, 2) decentralized formulation of the multi-agent risk allocation optimization problem using market-based method, and 3) introduction of Brent's method to tâtonnement-based resource allocation algorithm. The simulation result showed that MIRA can optimize the action sequence of the multi-agent system by optimally distributing risk. It achieved substantial speed-up compared to centralized optimization approach, particularly in a large problem.

7. ACKNOWLEDGMENTS

Thanks to Michael Kerstetter, Scott Smith and the Boeing Company for their support. Hirokazu Ishise and Yuichi Yamamoto provided valuable comments on economics. Thanks also to Shuonan Dong, Cristina Wilcox, and Masako Kishida for proofreading.

8. REFERENCES

[1] K. E. Atkinson. *An Introduction to Numerical Analysis, Second Edition*. John Wiley & Sons, 1989.

[2] R. Becker, V. Lesser, and S. Zilberstein. Decentralized mdps with event-driven interactions. In *Proceedings of AAMAS-04*, 2004.

[3] K. Bell, A. I. Coles, M. Fox, D. Long, and A. J. Smith. The application of planning to power substation voltage control. In *Proceedings of ICAPS Workshop on Scheduling and Planning Applications*, 2008.

[4] L. Blackmore. A probabilistic particle control approach to optimal, robust predictive control. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[5] L. Blackmore, H. Li, and B. C. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *Proceedings of American Control Conference*, 2006.

[6] L. Blackmore and M. Ono. Convex chance constrained predictive control without sampling. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2009.

[7] J. C. Jacobo, D. de Roure, and E. H. Gerding. An agent-based electrical power market. In *Proceedings of AAMAS-08: Demo Papers*, 2008.

[8] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[9] J. Marecki and M. Tambe. Planning with continuous resources for agent teams. In *Proceedings of AAMAS-09*, 2009.

[10] J. M. Martínez. Solving nonlinear simultaneous equations with a generalization of brent's method. *BIT Numerical Mathematics*, 20:501–510, 1980.

[11] D. Musliner, E. Durfee, J. Wu, D. Dolgov, R. Goldman, and M. Boddy. Coordinated plan management using multiagent mdps. In *AAAI Spring Symposium*, 2006.

[12] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synergy of distributed constraint optimization and pomdps. In *Proceedings of IJCAI-05*, 2005.

[13] M. Ono and B. C. Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *Proceedings of AAAI-08*, 2008.

[14] M. Ono and B. C. Williams. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Proceedings of 47th IEEE Conference on Decision and Control (CDC-08)*, 2008.

[15] M. Ono and B. C. Williams. Risk allocation for multi-agent systems using tâtonnement. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, 2009.

[16] A. Prékopa. The use of discrete moment bounds in probabilistic constrained stochastic programming models. *Annals of Operations Research*, 85:21–38, 1999.

[17] J. Tuinstra. *Price Dynamics in Equilibrium Models: The Search for Equilibrium and the Emergence of Endogenous Fluctuations*. Kluwer Academic Publishers, 2000.

[18] R. M. Turner and E. H. Turner. A two-level, protocol-based approach to controlling autonomous oceanographic sampling networks. *IEEE Journal of Oceanic Engineering*, 26, 2001.

[19] D. H. van Hessem. *Stochastic inequality constrained closed-loop model predictive control with application to chemical process operation*. PhD thesis, Delft University of Technology, 2004.

[20] H. Voos. Agent-based distributed resource allocation in technical dynamic systems. In *Proceedings of IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS 2006)*, 2006.

[21] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[22] F. Ygge and H. Akkermans. Power load management as a computational market. In *Proceedings of Second International Conference on Multiagent Systems*, 1996.

[23] E. R. Young. Unemployment insurance and capital accumulation. *Journal of Monetary Economics*, 51:1683–1710, 2004.