

Multi-Agent Computer Control System with Iterative Optimisation for Autonomous Agent Coordination

Field of the Invention

[0001] The present invention relates to multi-agent computer control systems and, more particularly, to iterative optimisation methods for coordinating autonomous agents in shared environments. The invention provides a structured optimisation framework that dynamically refines agent states using mathematical transformations, enabling efficient real-time decision-making. The system is applicable to autonomous vehicle coordination, robotic systems, distributed computing, and other multi-agent environments where optimised interaction between agents is required.

Background of the Invention

[0002] A Multi-Agent System (MAS) is a computational framework in which multiple autonomous entities, known as agents, interact within a shared environment to achieve individual or collective objectives. These agents are capable of acting independently, reasoning, and, in some cases, learning from their experiences. Due to their localised perception of the environment, agents must communicate and coordinate with one another to address complex problems effectively.

[0003] MAS is applied across various domains, including robotics, distributed computing, simulation, and smart grid management, where the diverse capabilities of agents enable efficient decision-making and problem-solving. One example is traffic management, where MAS can optimise vehicle flow at intersections. Each vehicle, whether autonomous or human-controlled, operates as an independent agent equipped with sensors and communication capabilities. As vehicles approach an intersection, they transmit their positions, speeds, and intended routes to a central traffic management system, which orchestrates vehicle movement in real-time. The system dynamically adjusts speeds and suggests entry times, minimising delays and preventing collisions. For instance, delaying one vehicle's approach may allow another to pass through the intersection without stopping, thereby reducing congestion. This coordination is often managed through an iterative process, where

real-time updates are continuously processed to optimise traffic flow. By leveraging MAS, traffic systems can operate with greater efficiency, scalability, and resilience, reducing reliance on traditional traffic signals.

[0004] Despite these advantages, optimising MAS presents significant challenges, particularly in environments where agents have conflicting objectives or operate with incomplete information. Efficient communication and coordination among autonomous agents become increasingly difficult as the complexity of interactions grows exponentially with the number of agents. The balance between cooperation and competition among agents requires sophisticated negotiation and conflict resolution strategies to align their actions towards common goals. Additionally, designing adaptive agents that learn and improve over time while maintaining system robustness against environmental changes adds further complexity.

[0005] A known approach to multi-agent coordination is described in US 2015/0284010 A1 (D1), which discloses a system for the shared control of semi-autonomous vehicles in multi-agent scenarios, particularly focusing on collision avoidance mechanisms. In D1, each vehicle functions as an agent that communicates its intended speed and direction to a central controller, which processes this information to establish permissible movement states and relational constraints. These constraints define the operational boundaries of each vehicle to prevent collisions and manage shared space occupation. The controller dynamically adjusts vehicle movements based on kinematic data and space occupancy constraints, thereby optimising vehicle trajectories within predefined limits.

[0006] While D1 introduces a structured method for vehicle coordination, its optimisation process is limited to enforcing static constraints and determining permissible states based on predefined rules. It lacks a robust iterative refinement process that would actively adjust system states over multiple steps to achieve a more optimal outcome.

[0007] As a result, D1 is limited in its ability to optimise multiple objectives in real-time, particularly in scenarios requiring complex trade-offs between agent interactions and this limitation makes D1 less adaptable to highly dynamic environments where

real-time multi-agent optimisation is required to balance competing objectives efficiently.

[0008] It is to be understood that, if any prior art information is referred to herein, such reference does not constitute an admission that the information forms part of the common general knowledge in the art, in Australia or any other country.

Summary of the Disclosure

[0009] There is provided herein a multi-agent computer control system configured to iteratively optimise the coordination of multiple computer agents by integrating relational constraints, agent state intentions, and a structured multi-dimensional optimisation approach. The system refines agent states through an iterative process that leverages mathematical transformations to identify optimal search directions and dynamically adjust system states in real time.

[0010] Existing approaches, such as that described in US 2015/0284010 A1 (D1), provide structured methods for vehicle coordination but suffer from significant limitations in optimisation. D1 primarily enforces static constraints and determines permissible states based on predefined rules, without an iterative refinement process that actively adjusts system states over multiple steps to reach an optimal outcome. The system in D1 does not incorporate a method for dynamically searching for improved system states beyond predefined constraints and lacks an adaptive approach to determine optimal search directions in a multi-variable space. Consequently, D1 struggles to optimise multiple objectives in real-time, particularly in scenarios requiring complex trade-offs between agent interactions. Furthermore, D1 does not leverage mathematical transformations that could improve computational efficiency and effectiveness. Instead, it relies on direct computation of vehicle movement constraints, rather than employing a structured, iterative approach that systematically refines agent states based on relational constraints and agent intentions. This limitation makes D1 less adaptable to highly dynamic environments where real-time multi-agent optimisation is required to balance competing objectives efficiently.

[0011] The present system overcomes these deficiencies by introducing a structured iterative optimisation process that systematically refines agent coordination through relational constraints, agent state intentions, and a structured multi-dimensional search approach. The system begins by initialising system states and relational state constraints, establishing the foundation for optimisation. It then receives state intentions from multiple computer agents, providing dynamic input that reflects real-time conditions. Based on this information, the system computes a combined set of permissible states, relational state constraints, and agent state intentions.

[0012] To enhance optimisation efficiency, the system iteratively optimises system states by converting between system states and Base_Equilibrium_Factor_Northogonal (BEFN) states. The BEFN states comprise a base point representing an initial reference state; equilibrium geometry, which includes equilibrium vectors and corresponding equilibrium states that improve outcomes; factor geometry, which includes factor vectors and corresponding factor states that are neutral and do not affect optimisation; and northogonal geometry, which includes northogonal vectors and corresponding northogonal states that depend on equilibrium and factor directions.

[0013] The system applies a mathematical transformation, wherein system states are calculated as $\text{system states} = \text{base point} + (\text{E vectors} * \text{E states}) + (\text{F vectors} * \text{F states}) + (\text{N vectors} * \text{N states})$, where the E, F, and N vectors are derived from relational state constraints and computer agent state intentions.

[0014] The iterative optimisation process proceeds through three sequential search steps: firstly, searching along equilibrium (E) geometry while ignoring factor (F) geometry, allowing optimal search directions to be identified without interference from neutral factors; secondly, searching along northogonal (N) geometry, refining state transitions in a manner that accounts for relational constraints; and thirdly, updating the base point, ensuring continued refinement and convergence toward an optimised solution.

[0015] Following these structured searches, the system calculates an optimised state based on the newly determined acceptable state and uses this optimised state to

dynamically control the computer agents. This structured process enables more efficient decision-making, prevents suboptimal solutions, and improves multi-agent coordination in real-time applications.

[0016] Unlike conventional gradient descent methods, such as those discussed in "Reducing Loss: Gradient Descent" (D2) retrieved from the Internet, 29 July 2024, <https://web.archive.org/web/20231209030939/https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent>, which simply compute a gradient and follow the steepest descent, the present system introduces multiple distinct optimisations beyond basic gradient calculations. While directional state refinement is utilised, it is implemented through a structured multi-vector transformation process rather than merely following a single gradient path. Specifically, the use of equilibrium, factor, and northogonal geometries provides additional search mechanisms that enable more adaptive and constraint-aware optimisation than a gradient descent approach alone. This structured transformation allows the system to avoid local minima, maintain robust agent coordination, and dynamically adjust the search space based on inter-agent relationships rather than treating the optimisation problem as a static mathematical function.

[0017] This multi-agent control system is particularly advantageous in highly dynamic environments, such as traffic coordination for autonomous vehicles, where agents must continuously adapt to shifting conditions. The structured optimisation process ensures efficient decision-making, reduces unnecessary computational overhead by focusing searches in pre-identified directions, and enhances overall system resilience and scalability in multi-agent environments.

[0018] Other aspects of the invention are also disclosed.

Brief Description of the Drawings

[0019] Notwithstanding any other forms which may fall within the scope of the present invention, preferred embodiments of the disclosure will now be described, by way of example only, with reference to the accompanying drawings in which:

[0020] Figure 1 illustrates a functional schematic of a multi-agent computer control system in accordance with an embodiment.

[0021] Figure 2 depicts a four-way intersection with up to 12 permissible movements.

[0022] Figure 3 depicts a three-way intersection with up to 6 permissible movements.

[0023] Figure 4 illustrates the steps of the gather process.

[0024] Figure 5 illustrates the steps of the convert process.

[0025] Figure 6 presents data structures used in the convert process.

[0026] Figure 7 presents data structures used for coalition formation.

[0027] Figure 8 illustrates the steps for calculating equilibrium geometry (E geometry).

[0028] Figure 9 illustrates the steps of the find class process.

[0029] Figure 10 illustrates the steps for searching along equilibrium geometry (E geometry).

[0030] Figure 11 illustrates the steps for calculating the optimisation objective value.

[0031] Figure 12 illustrates the steps of the expand class process.

Description of Embodiments

[0032] Figure 1 illustrates a multi-agent computer control system 100 adapted to optimise road intersection throughput for autonomous vehicles 106. In this embodiment, the autonomous vehicles 106 communicate with a central intersection controller 101, enabling system 100 to optimise intersection throughput by controlling the entry times of each autonomous vehicle 106 as it transits the intersection.

[0033] The intersection controller 101 comprises one or more processors for processing digital data and includes a memory device operably coupled thereto, configured to store digital data, including computer program code instructions. During operation, the processor fetches these instructions and associated data for interpretation and execution of the control functionality described herein.

[0034] These computer program code instructions may be logically divided into multiple computer program code instruction controllers. Similarly, each autonomous vehicle 106 may include respective processors and memory devices configured with computer program code instructions.

[0035] The autonomous vehicles 106 and the intersection controller 101 may communicate via a data network 109, such as a wireless data network. Figure 1 depicts each autonomous vehicle 106 transmitting its intentions 109 to the

intersection controller 101. These intentions may include the intended entry time and exit route of the intersection.

[0036] In other words, each autonomous vehicle 106 approaching the intersection communicates its intended entry time. In some embodiments, each autonomous vehicle 106 may also transmit its desired exit route to the intersection controller 101.

[0037] Given that large numbers of vehicles 106 may be transiting intersection 200, the controller 101 may group vehicles 106 into transit groups. For example, with reference to Figure 2, there may be 12 transit groups, each consisting of four approaches and three turn directions per approach. Each of these 12 transit groups may be treated as an independent computer agent that competes with other approaches but cooperates with turn directions within the same approach. The multiple transit groups constitute a multi-agent system that is computer-controlled within the terminology of this specification.

[0038] The system 100 may be configured with stored intersection logic 107, which may include permissible system states, relational state constraints, intersection cycle time, and intersection capacity for each movement within a cycle.

[0039] Within the context of traffic intersections, Figure 2 depicts a left-hand drive four-way intersection 200 with 12 permissible transit groups. Each transit group is characterised by four states per intersection cycle: (1) the time the transit group is using intersection 200, (2) the time the transit group is queueing to use the intersection while it is empty, (3) the residual number of vehicles that do not transit during their arrival cycle, and (4) intersection efficiency, defined as the number of vehicles transiting intersection 200 in each movement divided by the maximum capacity of the intersection 200 for that movement.

[0040] There are 48 system states for the intersection in Figure 2. These system states can be expressed mathematically as follows:

- a. P_{ij} : the time that the transit group with Approach i (1, 2, 3, or 4) and Turn j (S, L, or R) is using intersection 200;
- b. Q_{ij} : the time that the transit group with Approach i (1, 2, 3, or 4) and Turn j (S, L, or R) is queueing for the empty intersection 200;

- c. R_{ij} : the residual number of vehicles in the transit group with Approach i (1, 2, 3, or 4) and Turn j (S, L, or R) that do not transit intersection 200 during their arrival cycle;
- d. S_{ij} : the intersection efficiency for vehicles in the transit group with Approach i (1, 2, 3, or 4) and Turn j (S, L, or R), defined as the number of vehicles transiting intersection 200 in each movement divided by the maximum capacity of the intersection 200 for that movement.

[0041] The stored intersection logic 107 may further include relational state constraints. For example, with reference to Figure 2, a relational state constraint may prevent a vehicle turning right (R3) from simultaneously entering the path of another vehicle transiting straight through the intersection (S1). Some vehicles 106 may transit intersection 200 at the same time, such as S1 and S3 or S2 and S4, while others cannot, such as S1 and R3, due to conflicting paths. Certain transits that appear to conflict, such as S1 and L4, can be managed by system 100 without requiring vehicle-to-intersection communication.

[0042] The intersection logic may also include data about the intersection, such as intersection cycle time, denoted as T , and intersection capacity for movements, measured in vehicles per minute (or the average time for a single vehicle to move through the intersection).

[0043] The stored intersection logic 107 may further include intersection optimisation objective functions that operate on the system states. If there is only one optimisation objective function, standard methods are well known for determining the states that minimise this function, thereby identifying the optimal states, as described, for example, in "Gradient Descent Algorithm — a Deep Dive" [<https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21>]. The control system 101 applies when there is more than one optimisation objective function, allowing for multi-objective optimisation to balance competing factors in determining the most efficient system states.

[0044] The system 101 also functions when stored intersection logic 107 contains vector-valued choice functions instead of scalar-valued optimisation objective

functions. Any scalar-valued function can be converted into a vector-valued choice function by taking the first derivative with respect to state variables. However, not every vector-valued choice function represents a first derivative, making system 101 applicable beyond systems that rely solely on optimisation objective functions.

[0045] Minimising each of the following individual optimisation functions yields optimal states for their respective transit groups. Minimising the $Q \cdot Q$ terms reduces queueing time when the intersection is empty, while minimising the variance of intersection efficiency ensures that transit groups share intersection capacity equitably.

[0046] Collectively, the following optimisation objective functions model the intersection:

$$N_{ij} \cdot Q_{ij} \cdot Q_{ij} + \text{variance}(S),$$

where N_{ij} is the number of vehicles in each transit group $[i, j]$ arriving per minute during an intersection cycle,

S is the array of efficiencies $[S_{ij}]$, and

$\text{variance}(S)$ represents the variance from the mean of S .

[0047] Those skilled in the art will recognise that different optimisation objective functions may be used to achieve the goals of minimising queueing time and ensuring equitable intersection efficiency distribution.

[0048] This embodiment includes twelve computer agents and twelve intersection optimisation objective functions. Other embodiments may include different numbers of computer agents and optimisation objective functions.

[0049] Figure 3 illustrates an exemplar intersection 201 with three possible approaches and two possible turn directions per approach, assuming that all vehicles can transit the intersection during their arrival cycle. This intersection has six possible transit groups and twenty-four system states, of which six states satisfy $R_{ij} = 0$, leaving eighteen non-zero states.

[0050] Figure 1 shows the controller 101 implementing a gathering process 102. Figure 4 details the steps in gathering process 102, which may be executed in a central intersection controller or distributed across vehicles 106.

[0051] In step 301, gathering process 102 receives autonomous vehicle intentions 109, including the intended entry time and exit route for the intersection from vehicles 106. Step 301 calculates the number of vehicles in each transit group that will arrive during the intersection cycle using data received from vehicles 106. The number of vehicles in each transit group arriving per minute during an intersection cycle can be mathematically represented as N_{ij} . Step 301 may estimate this value based on information provided by a sample of vehicles 106 within communication range or historical data from the previous intersection cycle.

[0052] Step 301 also retrieves intersection data from store 107, which may include permissible system states, relational state constraints, cycle time, and the maximum intersection capacity in vehicles per minute for each transit group.

[0053] Step 301 further receives preferred choices for each transit group, either from store 107 or directly from vehicles 106. These preferences may be communicated as scalar-valued optimisation objective functions or vector-valued choice functions. If scalar-valued optimisation objective functions are provided, step 301 calculates the first derivative with respect to system states to convert them into vector-valued choice functions. Many computing libraries provide functions for calculating derivatives of symbolic expressions, for example, [<https://au.mathworks.com/help/symbolic/sym.diff.html>]. Those skilled in the art will understand that system 101 does not require functions to be stored in symbolic form but can compute approximate numerical derivatives as explained in [<https://au.mathworks.com/help/matlab/numerical-integration-and-differentiation.html>].

[0054] Step 302 integrates information from step 301, including vector-valued choice functions, into mathematical equations, which may be expressed in symbolic form as shown below. The exemplar intersection 201 in Figure 3 has eighteen non-zero system states, represented as P_{ij} , Q_{ij} , and S_{ij} , since $R_{ij} = 0$.

[0055] The relational system state constraints may be stored in symbolic mathematical form as follows, relative to the start of the intersection cycle time:

$$T - P_{11} - P_{23} - Q_{11} = 0$$

$$T - P_{13} - P_{31} - P_{23} - Q_{13} = 0$$

$$Q_{22} = 0$$

$$T - P_{23} - G_{13} - Q_{23} = 0$$

$$T - P_{31} - P_{13} - P_{23} - Q_{31} = 0$$

$$Q_{32} = 0$$

$$T - P_{22} = 0$$

$$T - P_{32} = 0$$

where $G_{13} = \text{smooth_max}(P_{11} + P_{33}, P_{13} + P_{31})$, and

$\text{smooth_max}(X, Y) = (X * \exp(k * X) + Y * \exp(k * Y)) / (\exp(k * X) + \exp(k * Y))$, with k set to 5 in this embodiment.

[0056] The relationship between intersection capacity and vehicle flow can be expressed as:

$$S_{ij} * P_{ij} * M_{ij} - T * N_{ij} = 0, \text{ for } (i, j) \text{ transit groups,}$$

where M_{ij} represents the maximum intersection capacity, expressed in vehicles per minute during cycle T , for vehicles in approach i making turn j , and

N_{ij} represents traffic density, expressed in vehicles per minute, for vehicles in approach i making turn j .

[0057] The optimisation objective functions to minimise queueing time and the variance of intersection efficiency (ensuring equitable efficiency distribution among transit groups) may be stored in symbolic mathematical terms as:

$Q_{11} * Q_{11} + Q_{13} * Q_{13} + \text{variance}(S)$, where S is the vector $[S_{ij}]$

$Q_{23} * Q_{23} + \text{variance}(S)$

$Q_{31} * Q_{31} + \text{variance}(S)$

[0058] There are three optimisation objective functions and six computer agents. Step 302 may compute symbolic first derivatives of the optimisation objective functions, resulting in vector-valued choice functions expressed as follows:

For the optimisation objective function $Q_{11} * Q_{11} + Q_{13} * Q_{13} + \text{variance}(S)$:

At the Q_{11} index: $2 * Q_{11}$

At the Q_{13} index: $2 * Q_{13}$

At the S_{ij} index for all i, j : $2/n * (S_{ij} - \mu)$, where n represents the number of multi-agent transit groups and μ is the mean of S , calculated as $\mu = \text{sum}(S_{ij}) / n$.

[0059] The vector-valued choice function is 0 for all indices not explicitly listed above.

[0060] Step 302 also stores the initial system states, derived either from the system states of the previous intersection cycle or an alternative estimate.

[0061] Step 303 determines whether the relational state constraints hold for the initial system states by evaluating the constraint equations from step 302. If the relational state constraints are satisfied, the initial system states are deemed feasible, and the process proceeds to step 304. Otherwise, the process advances to step 306.

[0062] Step 304 transmits the equations and feasible system states 113 via process 103 for conversion into a format suitable for process 104, which calculates the preferred intersection usage by multi-agents.

[0063] Step 305 receives results 115 from process 103. If results are not returned within a pre-specified time, step 305 continues using the initial system states. Step 305 then transmits the system states back to vehicles 106, specifying the time P_{ij} that vehicles can spend in the intersection, the time Q_{ij} that vehicles spend queueing at an empty intersection, and S_{ij} , the efficiency of intersection usage.

[0064] Step 306 is executed when the relational state constraints are not satisfied for the initial system states, meaning the intersection cannot function with those states. Step 306 then searches for feasible system states that satisfy the relational state constraints. Many well-established methods exist for efficiently conducting this search. In this embodiment, the search may be implemented using a standard constrained multi-variable optimisation process, minimising the distance between the initial and feasible system states while ensuring that equality and inequality constraints remain satisfied.

[0065] Step 306 then verifies whether the multi-variable optimisation process has returned feasible system states that meet all equality and inequality constraints. If feasible system states are found, the process proceeds to step 304. If no feasible system states exist, the process reports the absence of feasible states and triggers an intervention alert 307, notifying an appropriate authority that manual intervention is required.

[0066] In intersection 201, intervention alert 307 does not occur, as vehicle flow rates are lower than intersection capacity, preventing queues from persisting between cycles. However, in traffic intersection 200, intervention alert 307 may be triggered if flow rates exceed intersection capacity. In such a case, R_{ij} would be nonzero, necessitating an additional set of transit group choices. More generally, intervention alert 307 indicates that additional system states and agent choices are required to manage the multi-agent system.

[0067] Figure 1 illustrates the controller 101 implementing a convert process 103. Figure 5 details the steps involved in convert process 103, which may be executed in a central intersection controller or distributed across vehicles 106.

[0068] Convert process 103 receives equations and feasible system states 113 from step 304 of process 102. It converts these inputs into format 115 for use by process 104, which calculates BEFN states to determine preferred intersection usage by multi-agent transit groups. Process 104 is a more efficient algorithm for identifying preferred BEFN states than process 102 is for finding preferred system states, as it

operates on smaller arrays. Process 103 may also convert BEFN states back into system states.

[0069] Base_Equilibrium_Factor_Northogonal (BEFN) states provide a structured mathematical representation of system states by decomposing them into four distinct components: a base point (B), equilibrium geometry (E), factor geometry (F), and northogonal geometry (N). The base point represents an initial reference state from which optimisation occurs. The equilibrium geometry consists of equilibrium vectors and corresponding equilibrium states that improve system outcomes, typically aligning with primary optimisation objectives. Factor geometry consists of factor vectors and corresponding factor states that do not directly affect optimisation but provide neutral variations that allow for adaptable search pathways. Northogonal geometry consists of northogonal vectors and corresponding northogonal states that define directions influenced by both equilibrium and factor geometries while being perpendicular to equilibrium constraints.

[0070] Mathematically, BEFN states are defined by the equation:

$$\text{system states} = \text{base point} + (\text{E vectors} * \text{E states}) + (\text{F vectors} * \text{F states}) + (\text{N vectors} * \text{N states}),$$

where the vectors E, F, and N are derived from relational state constraints and computer agent state intentions.

[0071] The decomposition of system states into BEFN components enables more efficient multi-dimensional searching, reducing computational complexity by allowing searches to proceed along distinct geometrical transformations rather than an undifferentiated, high-dimensional space. This structured approach ensures that the optimisation process systematically refines system states in a way that aligns with inter-agent relationships while maintaining computational efficiency.

[0072] Those skilled in the art will understand that process 103 can be implemented through computer coding, computer chip micro-coding, or computer chip design to enhance computational efficiency.

[0073] The optimisation calculations involve continuous communication between gathering process 102 and find class process 104 via convert process 103.

[0074] Figure 6 illustrates the geometries and associated computer arrays in an embodiment comprising seven system and BEFN states. Array 501 is a standard computer array A, consisting of a number of memory locations that can be accessed as $A(i)$, where i takes values from 1 to 7 in Figure 6. Array A 501 is sometimes referred to as a vector because it has only one column. Array 502 is another computer array that implements the base point B. Arrays 503 to 509 form the columns of a computer array M that implements geometries as follows.

[0075] E geometry consists of a number $\dim E$ of E vectors, which in this embodiment are represented by 503 and 504, along with associated states 510 and 511. In this embodiment, $\dim E$ equals 2. F geometry consists of a number $\dim F$ of F vectors, represented by 505, along with associated states 512. In this embodiment, $\dim F$ equals 1. N geometry consists of a number $\dim N$ of N vectors, represented by 506, 507, 508, and 509, along with associated states 513, 514, 515, and 516. In this embodiment, $\dim N$ equals 4. The elements 510 to 516 may also be referred to as vector V, as they form a computer array with a single index variable.

[0076] These arrays are mathematically related by the formula $A = B + M$ multiplied by V, where the multiplication follows standard matrix multiplication such that $(M * V)_{i,j}$ represents the sum over k ($M_{ik} * V_k$) in the conventional notation, where row indices precede column indices. The E vectors and E states constitute E geometry, the F vectors and F states constitute F geometry, and the N vectors and N states constitute N geometry. The EFN geometries may be represented in a radial diagram 517, where the radial lines represent unit-length vectors and the bars represent the states.

[0077] Step 401 receives equations and feasible system states 113 from process 102. Step 401 also receives coalition information 111 from store 108, enabling system 101 to implement controls that adapt to the level of competition and cooperation among the multi-agents. The ability to use coalitions to regulate competition and cooperation is a unique feature of control system 101.

[0078] Figure 7 presents exemplar data structures for coalition information 111 in the embodiment of intersection 201. Data structure 601 represents a coalition comprising six computer agents acting cooperatively within a single group. Data structure 602 depicts a different coalition with three competing groups, where two agents cooperate in each of three groups. Data structure 603 represents six entirely independent groups, each competing with the others. The use of coalitions has been discussed in the literature, for example, in "Coalition Structure Generation: A Survey" [<https://www.sciencedirect.com/science/article/pii/S0004370215001198>]. Those skilled in the art will understand how more advanced information storage and processing techniques may be employed for managing interactions among multiple agents.

[0079] Step 402 assigns the vector of feasible system states as the base point B. It also determines northogonal directions by calculating the gradients of all equality relational state constraint equations to generate a set of northogonal vectors. Northogonal refers to a combination of perpendicularity to the tangent (termed normal in mathematics) and orthogonality (meaning that the inner product of vectors is either 0 or 1). It is well known in mathematics that the gradient of a level set function is perpendicular to the tangents of the level set. The number of northogonal vectors is denoted as dimN. In the embodiment 201, there are 14 northogonal directions.

[0080] Since all northogonal directions are nonzero, the northogonal vectors can be converted into orthogonal terms, ensuring that each vector has unit length and that the inner product between different vectors is zero.

[0081] Process 403 calculates equilibrium geometry, which is orthogonal to northogonal geometry. Figure 8 illustrates the steps in process 403.

[0082] Step 404 processes the results from step 403 and calculates factor geometry by determining a set of vectors that are orthogonal to both N geometry and E geometry. This calculation follows a linear algebra method, such as performing singular value decomposition on the matrix that projects perpendicular to N and E geometry. Various computational libraries provide functions for singular value

decomposition, for example, [<https://au.mathworks.com/help/matlab/math/singular-values.html>].

[0083] Step 405 sends the results to process 104 and facilitates conversion between system states and BEFN states for fixed B, E, F, and N as required. Step 406 receives the results from process 105 and communicates them back to process 102.

[0084] Figure 8 illustrates process 403. Step 701 initiates loops and sets the initial number of E vectors and states to zero. Step 702 checks whether there are any groups in coalition 111 that have not been processed or whether any unused choices remain in store 709. If no additional groups or choices remain, process 403 terminates and process 404 begins. If there are unused groups or choices, the process proceeds to step 703.

[0085] Step 703 retrieves the next group of choices from coalition 111 or from store 709. Step 704 calculates a vector-valued choice function for the group and ensures that it is orthogonal to all previously computed choice functions in process 403. In the embodiment, step 704 computes the average of the vectors in the group and then determines the projection that is orthogonal to prior choice functions. Those skilled in the art will recognise that alternative methods may be used to derive a single vector-valued choice function that represents the coalition group.

[0086] Step 705 calculates the norm of the projected vector-valued choice function from step 704. If the norm exceeds a specified threshold, accounting for computational precision, the vector-valued choice function is normalised to unit length and added to store 708, after which the process returns to step 702. If the norm is below the threshold, step 706 begins. If all vectors in the group have norms below the threshold, step 706 begins.

[0087] Step 706 removes one vector from the group and adds it to store 709. The vector to be removed is the one with the smallest inner product relative to all other vectors in the group. Step 706 then recalculates a vector-valued choice function for the reduced group and ensures that it remains orthogonal to all prior choice functions in process 403. The process then returns to step 705 to reassess the norm. If the norm exceeds the specified threshold, the vector-valued choice function is added to

store 708, and the process returns to step 702. If the norm remains below the threshold, step 706 is repeated. If no further vectors can be removed, step 707 begins.

[0088] Step 707 adds randomly selected vectors until the number of E state vectors in store 708 is at least equal to the number of groups in the coalition.

[0089] Find class process 404 searches for optimal BEFN states through three separate processes to enhance efficiency. Figure 9 illustrates the steps in process 404 for the embodiment. Those skilled in the art will understand that alternative steps may achieve the same results.

[0090] Step 801 retrieves search parameters 112 from store 109, allowing system 101 to implement controls that adapt to the type of optimisation required, as described below. The ability to modify optimisation settings is a unique feature of control system 101. Those skilled in the art will understand that a range of information may be stored and utilised for search parameters.

[0091] Step 801 initialises the search by setting a maximum search time counter, defining a logical termination value initially set to false, and clearing store 807. Step 802 determines whether the search should terminate, either because the search time counter has been exceeded or because the logical termination value is true. If the search should terminate, step 802 selects the search results from store 807 with the lowest values of the optimisation objective function 1004 and returns the BEFN states 116 to the next step in process 100, namely process 105. If the search is to continue, the process moves to step 803.

[0092] Step 803 initiates a search along E geometry while keeping the base point B fixed. In embodiment 201, the search minimises the expression $(1 + \text{constraint_penalty}) * (1 + S_m) * (\text{constraint_penalty} + S_m + \text{penalty term for F geometry})$. Step 803 also establishes a maximum search time counter and logical termination value.

[0093] Steps 804 and 805 constitute the search process. Step 804 checks whether the search should terminate because the search time counter has been exceeded or

the termination condition is met. If termination is required, search results are passed to step 806. Otherwise, the process proceeds to process 805.

[0094] Step 806 updates the base point B and stores it in store 807. Step 806 sets the logical termination value based on whether the change in the optimisation objective function between consecutive iterations of step 806 falls below a threshold determined by computational precision. The process then returns to step 802.

[0095] Process 805 consists of the steps shown in Figure 10. Step 901 may use a standard optimisation algorithm to search along E geometry. In embodiment 201, the algorithm employed is a gradient search using interior points.

[0096] Step 902 prepares to calculate the optimisation objective function value. Step 903 initiates a search along N geometry, setting a search time counter and initially defining the logical termination value as false.

[0097] Step 904 assesses whether the search should terminate, either because the search time counter has been exceeded or because the logical termination value is true. If termination is required, search results are passed back to step 804. Otherwise, the process proceeds to step 905.

[0098] Process 905 employs a standard optimisation algorithm to search along N geometry. In embodiment 201, the algorithm is a gradient search using interior points. Step 906 verifies that all constraints are met by converting BEFN states back into system states and evaluating constraints. The resulting constraint values are passed to step 907.

[0099] Figure 11 illustrates search process 907. Step 907 sets the logical termination value based on whether the change in the optimisation objective function between consecutive iterations of step 907 is below a threshold determined by computational precision. The process then returns to step 904.

[0100] Step 1001 serves as the entry point for search process 907, receiving search parameters 112 from store 109 and constraint values from step 906. In the embodiment of intersection 201, parameters 112 include two variables, num_m and num_nm, along with a search scale s.

[0101] Step 1002 computes the matrix product between vector-valued choice function 1006 and the vectors in E geometry 1007. The matrix size of the vector-valued choice functions is num_choices multiplied by dim_num , and the matrix size of E geometry is dim_num multiplied by dimE . As a result, the matrix product 1008 has num_choices rows and dimE columns.

[0102] Step 1003 computes the singular value decomposition of matrix product 1008 using an existing computational software routine capable of sorting the singular values in descending order, ensuring that all singular values are non-negative. Embodiment 1008 contains two singular values 1009.

[0103] Step 1004 calculates S_{nm} as the sum of the num_nm highest singular values and S_m as the sum of the num_m lowest singular values.

[0104] In embodiment 201, step 1005 returns the sum of S_m and the unsatisfied constraint values, ensuring that the search minimises variation in choices along the equilibrium geometry. In alternative embodiments, the search may minimise S_{nm} , S_m , and constraints collectively.

[0105] Find process 104 does not search along all F geometry and may therefore not identify every BEFN state that optimises multiple agent choices. This is a unique feature of method 101 that improves efficiency.

[0106] In embodiment 201, different coalitions 108 produce different E geometries and F geometries. Coalition 603 has dimE equal to 4 and dimF equal to 0, resulting in only one optimal BEFN state. This BEFN state is communicated as 116 to process 105 and then directly forwarded as 118 to process 103, which converts BEFN state 118 into system state 114 and transmits it via process 102 to vehicles 106. Coalition 601 has dimE equal to 1 and dimF equal to 3, meaning it may generate more than one optimal BEFN state. Process 105 expands the set of BEFN and system states into a larger set.

[0107] Figure 12 illustrates how process 105 can return multiple system states. One embodiment of process 105 utilises Markov Chains, as described in "Introduction to Markov Processes" [<https://ocw.mit.edu/courses/res-6-012-introduction-to-probability-spring-2018/resources/introduction-to-markov-processes/>].

[0108] Step 1101 receives BEFN states 116 from process 104 and input 117 from store 110. Step 1102 establishes a Markov Chain or a similar probabilistic process. Step 1103 checks whether the Markov Chain process has completed. If complete, the process advances to step 1109. If not complete, the process moves to step 1104, which may implement a Markov process on F geometry.

[0109] Step 1104 generates pseudo-random F states using information 117. In one embodiment, this calculation employs a standard uniform pseudo-random number generator scaled between 0 and 1. However, any pseudo-random number generator could be used.

[0110] Step 1105 calculates a new base point C by adding the pseudo-random F states, multiplied by F vectors, to the base point B in input 116, using the calculation:

$$C = B + (F \text{ vectors multiplied by pseudo-random F states}).$$

[0111] Step 1105 then transmits base point C, E geometry, F geometry, and N geometry to process 104 to compute an additional set of BEFN states 116.

[0112] Step 1106 converts BEFN states 116 into system states and stores them in 1108, along with an index identifying each sample element in the Markov Chain. The first element is identified as 1, with the n-th element assigned n.

[0113] Step 1109 summarises the system states stored in 1108. In the embodiment, the summary calculation determines the simple average of the system states, though this is unlikely to satisfy all constraints. Consequently, the summary is sent back to process 104 to determine the closest BEFN states. Step 1109 then communicates results 118 through convert process 103 to gather process 102. Those skilled in the art will understand that other summary measures, such as cluster means, may also be employed.

[0114] The Markov Chain modelling method may also be applied to process 102 in cases where probabilistic choices replace deterministic choices. This would necessitate additional indices to track elements within the Markov Chains for each probabilistic input and supplementary summarising processes to report the outputs.

[0115] The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that specific details are not required in order to practise the invention. Thus, the foregoing descriptions of specific embodiments of the invention are presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed as obviously many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, thereby enabling others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the following claims and their equivalents define the scope of the invention.

Claims

1. A multi-agent computer control system comprising a controller executing computer program code instructions, the controller being in operable communication with a plurality of computer agents, wherein the controller is configured to:

initialise system states and relational state constraints;

receive state intentions from the plurality of computer agents;

compute a combined set of permissible states, relational state constraints, and computer agent state intentions;

iteratively optimise system states by:

converting between system states and

Base_Equilibrium_Factor_Northogonal (BEFN) states, wherein the BEFN states comprise:

a base point representing an initial reference state;

equilibrium geometry, which includes equilibrium vectors and corresponding equilibrium states that improve outcomes;

factor geometry, which includes factor vectors and corresponding factor states that are neutral and do not affect optimisation; and

northogonal geometry, which includes northogonal vectors and corresponding northogonal states that depend on equilibrium and factor directions;

calculating system states = base point + (E vectors * E states) + (F vectors * F states) + (N vectors * N states), where the E, F, and N vectors are derived from relational state constraints and computer agent state intentions;

searching for a new acceptable state in three sequential steps:

firstly, searching along E geometry while ignoring F geometry;

secondly, searching along N geometry; and

thirdly, updating the base point to refine the state selection;

calculating an optimised state based on the newly determined acceptable state; and

controlling the computer agents based on the optimised state.

2. The system as claimed in claim 1, wherein the iterative optimisation process employs explicit control parameters to differentiate between collaborative and competitive equilibrium vectors and factor vectors, and wherein the search is restricted to equilibrium vectors to enhance efficiency.
3. The system as claimed in claim 1, wherein the initial system states are derived from the final system states of a previous optimisation cycle.
4. The system as claimed in claim 1, wherein the iterative optimisation process terminates upon reaching a predefined elapsed time threshold.
5. The system as claimed in claim 1, wherein the iterative optimisation process includes determining whether a search direction at a subsequent point is more optimal than at the current point by computing the alignment between equilibrium search directions and the state intentions of the computer agents, wherein the alignment is calculated using singular value decomposition of the matrix product of equilibrium search directions and the agents' state intentions.
6. The system as claimed in claim 5, wherein the controller identifies a lower combined alignment value as being more optimal.
7. The system as claimed in claim 1, wherein the computer agents comprise groups of autonomous vehicles in operable communication with an intersection controller via a data network.
8. The system as claimed in claim 7, wherein the optimised state is determined to maximise intersection throughput.

9. The system as claimed in claim 7, wherein the optimised state is determined to minimise vehicle wait times at the intersection.
10. The system as claimed in claim 1, wherein the controller dynamically adjusts the base point during the iterative optimisation process based on system states derived from a previous optimisation cycle.
11. The system as claimed in claim 1, wherein the northogonal geometry is derived by calculating gradients of equality relational state constraints to form northogonal vectors that define feasible state transitions.
12. The system as claimed in claim 1, wherein the equilibrium geometry is determined based on vector-valued choice functions representing the collective intentions of the computer agents.
13. The system as claimed in claim 1, wherein the factor geometry is computed using singular value decomposition of a matrix that projects perpendicular to equilibrium and northogonal geometries.
14. The system as claimed in claim 1, wherein the optimisation process applies an iterative update mechanism that adjusts system states based on relational state constraints and agent intentions until a stable optimised state is achieved.
15. The system as claimed in claim 1, wherein the system implements constraint validation after each optimisation iteration by converting BEFN states back into system states and evaluating compliance with relational state constraints.
16. The system as claimed in claim 1, wherein the controller applies a multi-step search process that first searches along equilibrium geometry, then along

northogonal geometry, and finally updates the base point to refine system state selection.

17. The system as claimed in claim 1, wherein the system uses an intervention alert to indicate when no feasible optimised state exists, requiring external input to resolve state conflicts.

18. The system as claimed in claim 1, wherein the relational state constraints include conditions preventing conflicting agent actions, such as simultaneous entry of vehicles into a restricted zone.

19. The system as claimed in claim 1, wherein the system processes multi-agent state intentions using a distributed computational framework.

20. The system as claimed in claim 1, wherein the agents intentions are different functional objectives in multi-objective optimisation.