A Scheme for Coordinating Multi-robot Planning Activities and Plans Execution

Rachid Alami, Félix Ingrand, Samer Qutub

A Scheme for Coordinating Multi-robot Planning Activities and Plans Execution

Rachid Alami and **Félix Ingrand** and **Samer Qutub** [1]

Abstract. We present and discuss a generic scheme for multi-robot cooperation based on an incremental and distributed plan-merging process. Each robot, autonomously and incrementally builds and executes its own plans taking into account the multi-robot context. The robots are assumed to be able to collect the other robots plans and to coordinate their own plans with the other robots plans to produce "coordinated plans" that ensure their proper execution.

We discuss the properties of this cooperative paradigm (coherence, detection of dead-lock situations, ...) , how it "fills the gap" between centralized and distributed planning and the class of applications for which it is well suited.

We finally illustrate this scheme through an implemented system which allows a fleet of autonomous mobile robots to perform load transfer tasks in a route network environment with a very limited centralized activity and important gains in system flexibility and robustness to execution contingencies.

1 Introduction

In the field of multi-agent cooperation, we claim that agents must be able to plan/refine their respective missions, taking into account the other agents plans as planning/refinement constraints, and thus producing plans containing coordinated action that ensure their proper execution.

This is particularly true for autonomous multi-robot applications and, more generally, when the allocated goals cannot be directly "executed" but require further refinement, because the robots act in the same physical environment and because of the multiplicity of uncertainties.

Let us assume a set of autonomous robots, which have been given a set of partially ordered goals. This could be the output of a central planner, or the result of a collaborative planning process. One can consider this plan elaboration process finished when the obtained goals have a sufficient range and are sufficiently independent to cause a substantial "selfish" robot activity. However, each robot, while seeking to achieve its goal will have to compete for resources, to comply with other robots activities. Hence, several robots may find themselves in situations where they need to solve a new goal interaction leading to a new goal/task allocation scheme.

In such context, planning and plan coordination can be classified along different strategies.

Global versus local. When one plans actions for a fleet of mobile robots, one can consider the whole fleet or limit the planning scope to the robots "involved" in the considered resources. Indeed, it seems to be rather inefficient to take into account all the robots present on the field for any local decision which involves only a subset of robots. However, this global versus local tradeoff is only possible when dealing with a "properly sized" environment. If the number of exclusive resources (such as spatial resource) is more or less equal to the number of robots, conflict resolution will, by propagation, involve the whole fleet. On the other hand, if the environment is "properly sized", conflicts remain local, and the solutions are negotiated locally without disturbing the unconcerned robots.

Complete versus incremental. Similarly, one can limit the scope of the planning and the plan coordination in time. When a mission (i.e. a set of goals) is sent to a robot, it can plan and coordinate the whole mission. But considering the execution hazards, and the inaccuracies with which one can forecast at what time such and such contingent actions will end, it seems to be inefficient (not to say a waste of time and resources) to plan too far ahead. The plan coordination process should be performed incrementally to avoid over constraining the other robots plans and to minimize the execution failures of the already coordinated plans.

Centralized versus distributed. This last aspect of the planning and plan coordination problem is where the planning and plan coordination should take place, on a central station or on board the robots. Centralized versus distributed does not change the computing complexity of the treatment. However, in a centralized approach, all the data (which are mostly local) need to be sent to the central station, and therefore require a more reliable communication channel with higher bandwidth between the robots and this central station.

The approach we have chosen may be classified as *local*, *incremental* and *distributed* [3]. However, when the situation imposes it, our paradigm may "evolve" dynamically towards a more centralized and global form of planning [13].

After this introduction which makes a general presentation of our approach, and situates it in the general multi-robot planning debate, we present the related work in section 2. We shall introduce a more formal presentation of the Plan Merging Paradigm (PMP) and its operators in section 3. Section 4 briefly presents the multi-robot application on which we tested and validated the PMP.

[1] LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse CEDEX 04, France. E-mail: {rachid,felix,sam}@laas.fr. Authors list in alphabetical order.

2 Related work

While several generic approaches have been proposed in the literature concerning goal decomposition and allocation (Contract Nets [16], Partial Global Planning [4], distributed search [5], negotiation [10, 7, 14], motivational behaviors [12, 6]), cooperation for achieving independent goals have been mostly treated using task-specific or application-specific techniques [11, 17]

We argue that there is also a need for generic approaches to perform plan coordination. One may introduce the notion of traffic rules or more generally the "social behaviors" [15] to avoid as much as possible conflicts and to provide predefined solutions to various well known situations. However, this cannot be considered as a general and applicable answer to the various multi-agent problems.

Our scheme provides and guarantees a coherent behavior of the robots in all situations (including the avalanche of situations which may occur after an execution failure) and a reliable detection of situations which call for a new task distribution process.

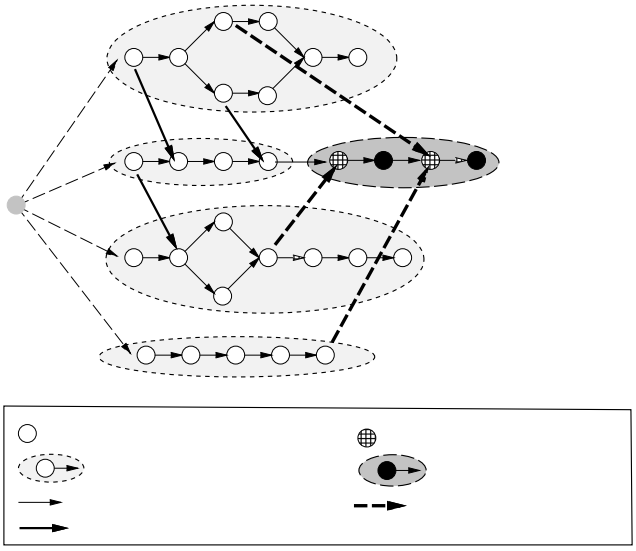

Figure 1. Robot 2 performs a Plan-Merging Operation.

3 Presentation of the PMP

Let us assume that we have a set of autonomous robots and a central station which, from time to time, sends goals to robots individually. Whenever a robot R_i receives a new goal G_i^j, it elaborates an *Individual Plan* (IP_i^j) which takes as initial state the final state of the current plan. Each robot processes sequentially the received goals. Doing so, it incrementally appends new actions to its current plan.

However, before executing any plan step, a robot must ensure that it is valid in the multi-robot context, i.e. all potential conflicts with the other robots plans are considered. We call this operation *Plan Merging Operation* (PMO) and the resulting plan a *Coordinated plan* (i.e. plan valid in the current multi-robot context). Such a *Coordinated Plan* (CP_i) consists of a sequence of actions and *execution events* to be signaled to other robots as well as *execution events* that are planned to be signaled by the other robots. Such *execution events* correspond to temporal constraints between actions involved in the different coordinated plans.

At any moment, the temporal constraints between all the actions included in the union of all the coordinated plans ($GP = \bigcup_k CP_k$) constitute a *directed acyclic graph* [3] which is a snapshot knowledge of the current situation and its already planned evolution (Fig. 1).

3.1 The PMO and its results

When R_i receives its j-th goal G_i^j, it elaborates a plan IP_i^j which achieves it; then it performs a PMO under mutual exclusion, in order to prevent simultaneous modification of GP: it collects the coordinated plans CP_k of the robots which may interfere with IP_i^j, and builds their union $GP = \bigcup_k CP_k$. The insertion of IP_i^j in the global plan GP, if it succeeds, adds temporal order constraints to actions in IP_i^j and transforms it into a coordinated plan CP_i. The out-coming CP_i is feasible in the current context, and does not introduce any cycle in the resulting GP.

However a PMO performed by R_i may fail because the final state of at least another robot R_k (as specified in GP) forbids R_i to insert its own plan IP_i^j in GP. Let us call $Pred_i = \{..R_k..\}$ the set of all such robots. In this case, R_i defers its PMO and waits until at least one of the robots in $Pred_i$ has performed a new successful PMO which may possibly change the world attributes preventing the insertion of IP_i^j. Hence, we introduce, when necessary, temporal order relations between the different plan-merging activities.

In addition to *execution events*, events elaborated by the PMOs and which allow the robots to synchronize their plans, we define *planning events*, events which occur whenever a robot performs a new successful PMO. The temporal relations between robots plan-merging activities are maintained by each robot R_i in a data structure called *Planning Dependency Graph* PDG_i.

The *Planning Dependency Graph* serves to manage PMOs order (when necessary) as well as to detect *waiting cycles* corresponding to "Planning Deadlock Situations". The detection of deadlocks during the coordination phase allows execution deadlocks to be anticipated and avoided where "backtracks" are not always possible or induce inefficient maneuvers.

3.2 Dependency Graph Construction

This section focuses on the incremental distributed construction of the Planning Dependency Graph PDG_i and its constraints propagation mechanism.

When R_i starts a new PMO, $Pred_i$ is set to the empty list. If the insertion of IP_i^j in GP succeeds, R_i signals a *planning event* to all robots in $Succ_i$ [2] and clears its current graph PDG_i.

If the insertion has failed, R_i determines $Pred_i$ and checks if it induces planning dependencies which produce cycles in PDG_i.

[2] We call $Succ_i$ the set of robots that are directly blocked by R_i.

- In such case, a *planning deadlock situation* is detected which means that the given goals are interdependent and cannot be treated simply by insertion, but need to be handled in a single planning step.
- If the newly established *planning dependencies* do not introduce any cycle in PDG_i, R_i transmits PDG_i to $Pred_i$.

When the robot R_k receives PDG_i from R_i, R_k adds it to its own Dependency Graph PDG_k and propagates this new information to all robots in $Pred_k$. R_k is sure that the received PDG_i can be merged with PDG_k without creating any cycle[3].

3.3 Deadlock Resolution Strategy

The deadlock resolution strategy that we present is based on a cooperative scheme. We assume that all robots are equipped with a multi-robot planner[4] which can be used, when necessary, for an arbitrary number of robots.

Let us call DL_i^j the set of robots involved in a cycle detected by R_i. When detecting a cycle, R_i has the necessary information in PDG_i to elaborate and validate a plan for all blocked robots in DL_i^j. Note that the blocked robots are unable to add any new executable action to their current coordinated plans CP_k. Therefore, if nothing is done, they will come to a complete stop when their plans CP_k has been completed.

To solve the deadlock, R_i becomes the local coordinator (noted R_i^{LC}) for all robots in DL_i^j. To do so, it makes use of its *Local Multi-robot Planner* that will take explicitly, in one planning operation, the conjunction of goals of the blocked robots. This fact will be represented in the Dependency Graph PDG_i as a *Meta-Node* that includes all robots in DL_i^j.

The local coordinator R_i^{LC} must find a multi-robot solution (Sol_i^j), if it exists, to the conjunction of goals. This solution is represented by a *lattice* whose nodes are high level actions to be performed to break the cycle and whose arcs are "synchronization events" between these actions. Once the solution found, R_i^{LC} tries to insert Sol_i^j in $GP = \bigcup CP_k$[5].

- If the insertion of Sol_i^j succeeds, R_i^{LC} sends to the robots in DL_i^j their plans and each robot in DL_i^j recovers its initial planning and plan-merging autonomy.
- If the insertion fails, this means that the final state of at least one robot (not included in DL_i^j) forbids R_i^{LC} to validate Sol_i^j. R_i^{LC} determines $Pred_i^{LC}$ and verifies that these newly established constraints do not introduce any cycle in PDG_i. In such case, R_i^{LC} defers its PMO, transmits PDG_i to all robots in $Pred_i^{LC}$ and waits until one of them has performed a new PMO.
  If a new cycle DL_i^{j+1} is detected, R_i^{LC} generates a new *Meta-Node* containing the union of DL_i^j and DL_i^{j+1} and recursively restarts the same process , acting as a coordinator of a greater set of robots.

[3] If such cycle existed, R_i would have discovered it.

[4] Note that it is not strictly necessary to have a multi-robot planner on each robot. A unique multi-robot planner, installed somewhere on the network (at the central station for instance), is sufficient to ensure a correct behavior of the system. The main point, here, is that our scheme is able to determine, in a conservative and incremental way, the set of robots involved in a deadlock and to invoke the multi-robot planner on the set of concerned robots without systematically taking into account all the robots.

[5] GP is the set of current coordinated plans CP_k of the robots which are not involved in DL_i^j

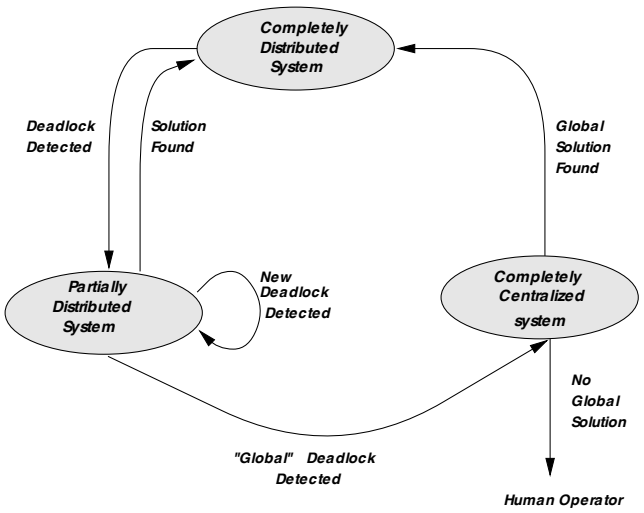

Figure 2. Progressive transition to a more global scheme

Note that we may imagine many parallel deadlocks which do not interfere and which are solved independently. At the same time, we may have some complicated situations where the *Meta-Node* grows up until the inclusion of the whole system transforming momentarily our distributed system to a completely centralized one (fig. 2).

3.4 Deadlock Resolution Example

To illustrate our deadlock resolution strategy, we treat a relatively complex situation where four robots evolve in a constrained space.

- R_0 (respectively R_3) is blocked by R_1 (respectively R_2) and thus waits for planning event from R_1 (R_2) to start a new PMO (Figure 4A) (Figure 3A).
- while performing a new PMO, R_1 (respectively R_2) detects a cycle DL_1^0 (DL_2^0) in its PDG_1 (PDG_2) involving R_0 and R_1 (R_3 and R_2). So, R_1 (R_2) becomes the local coordinator R_1^{LC} (R_2^{LC}) of DL_1^0 (DL_2^0) and tries to find a Multi-Robot plan Sol_1^0 (Sol_2^0) for the missions of R_1 and R_0 (R_3 and R_2) (Figure 4B, 4C)(Figure 3B,3C).
- Sol_1^0 and Sol_2^0 are dependent and thus cannot be inserted in GP without introducing a cycle. R_2^{LC} becomes the local coordinator of both R_3 and R_1^{LC} and thus by transitivity it becomes also the coordinator of R_0. R_2^{LC} generates and validates Sol_2^1 in GP[6] (Figure 4D)(Figure 3D).
- Sol_2^1 is distributed to the concerned robots for execution (Figure 4E).

After solving the deadlock situation, each robot finds its initial planning/coordination autonomy.

[6] Sol_2^1 is Multi-Robot plan that achieves all the given missions

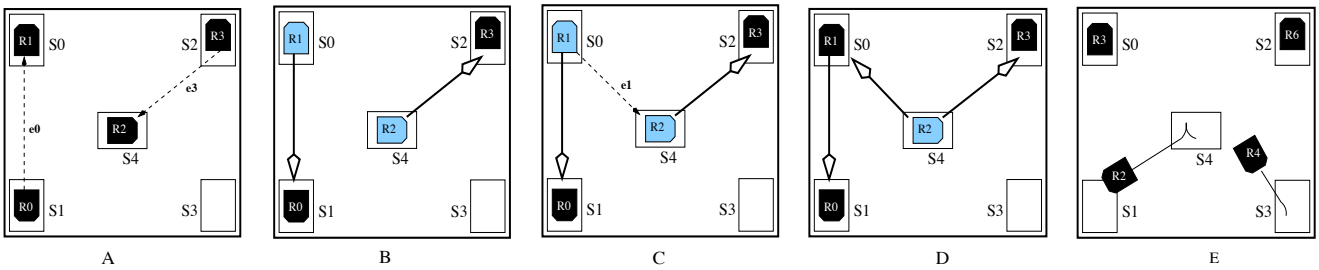

Figure 4. An Example of deadlock resolution strategy by Meta Node expansion involving four robots. The robots in gray are the local coordinators of the local deadlocks.

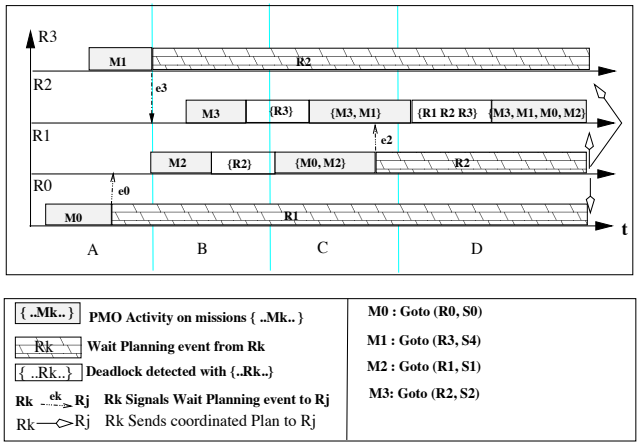

Figure 3. The evolution of PMO states in time.

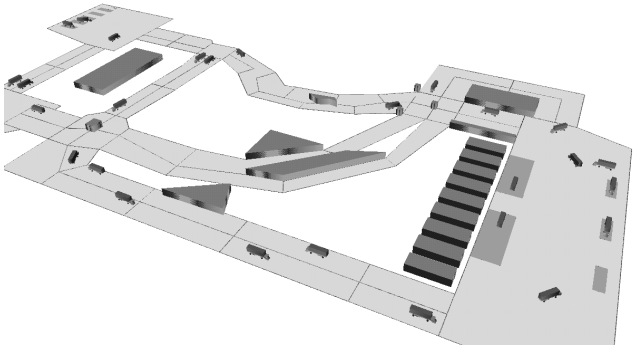

Figure 5. Simulation with 27 autonomous mobile robots.

Figure 6. The 3 Hilare robots executing their coordinated plans.

4 Application to a Fleet of Autonomous Mobile Robots

We have applied the Plan-Merging Paradigm in the framework of a project which deals with the control of a large fleet of autonomous mobile robots for the transportation of containers in harbors, airports and railway environments [2].

In such context, the dynamics of the environment, the impossibility to correctly estimating the duration of actions (the robots may be slowed down due to obstacle avoidance, and delays in load and un-load operations, etc..) prevent a central system from elaborating efficient and reliable detailed robot plans.

The use of the Plan-Merging paradigm allowed us to deal with several types of conflicts in a general and systematic way, and to limit the role of the central system to the assignment of tasks and routes to the robots (without specifying any trajectory or any synchronization between robots) taking only into account global traffic constraints.

The robots are fully autonomous; they only receive high level goals from time to time. They elaborate their own motion plans. Plan Merging is performed at two levels: the first level deals with spatial resource use (cells) while the second level deals with trajectory synchronizations. This hierarchy authorizes a "light" cooperation, when possible, and a more detailed one, when the situation is more intricate.

The overall system has been implemented, using the architecture and tools presented in [1, 9, 8] and has been run in "close to real world" simulations (fig. 5) involving a large number of robots (up to 30) as well as on real lab robots in a constrained environment (fig. 6).

We have conducted several experiments on different environment topologies. The system proved to be really efficient, with reasonable communication bandwidth requirements and effective ability to deal with non-trivial situations[3, 2].

The whole process showed effective incremental behavior. A robot may "enter" into coordination process concerning several robots, and "leave" it after a while, without the need

to maintain a unique representation of the global plan. Its construction as well as its execution are performed in a distributed and synchronized manner.

We discuss here below some aspects that we have drawn from our experience in the effective use of the Plan-merging paradigm.

Planning before or during a PMO: The choice between this two possibilities depends mainly on the application and on the extent of plans which have to be merged.

Note also that merging plans consisting in long sequences of actions may induce a great number of constraints for the future PMOs. This is again application dependent. For example, in traffic applications, it is certainly better to limit the range of the inserted plan in order to allow a smooth traffic.

Satisfying real-time constraints: Note that the paradigm we propose does not impose any constraints on the time necessary for planning, performing a PMO or executing an action.

Indeed, in the general case, planning time cannot be bounded. In any case, the execution may continue, until the coordinated plan is completely executed, while planning or PMO is performed.

This is why robots synchronization is based on events as perceived and produced by robots along their execution and not on a numerical estimation of the duration of actions of other operations performed by robots.

Accounting for execution failures: The Plan-Merging paradigm is also robust to execution failures. Indeed, as execution is synchronized through event produced by the robots, when a robot fails in the execution of one of its actions, it is able to inform robots which ask for the occurrence of events it is supposed to produce, that such events will never occur.

This information may cause other robot plans to fail. All robots which have a "broken" coordination plan will rebuilt their state and try a PMO again.

Depending on the constraints imposed by an event which will not occur, a cascade of plan failures may occur. This may cause a brutal increase of PMO activities with several robots trying to perform a PMO at almost the same time, but the system will be maintained safe thanks to the properties discussed earlier (guarantee of always having a valid global plan and of detecting deadlocks or situations where a PMO should be deferred).

5 Conclusion

The effectiveness of the Plan-merging paradigm has already been discussed and illustrated through the implementation of a system involving up to 30 simulated mobile robots. It has also been implemented on a set of 3 real robots in a laboratory environment[2].

The Plan-merging paradigm is a well suited paradigm to multi-robot applications with loosely-coupled tasks. However, even if an application is designed to ease robots interaction, one cannot guarantee in the general case that tightly-coupled tasks will never happen. For example, the robots may find themselves in intricate situations simply because of an unknown obstacle placed in a critical place. This is why the plan-merging paradigm has been extended such that the system is able to efficiently exploit the tasks decoupling, but is also able to detect and solve transient "puzzle-like" situations.

We have presented here a set of extended operators and associated mechanisms which allow not only to detect but also to solve situations where the robots goals are tightly coupled. This extension is done for the sake of completeness. The operators permit a coherent management of the distributed planning and coordination processes as well as a progressive transition to more global schemes which may even "degrade" to a unique and centralized planning activity.

REFERENCES

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, 'An architecture for autonomy', *International Journal of Robotics Research*, **17**(4), 315–337, (April 1998).

[2] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, 'Multi Robot Cooperation in the Martha Project', *IEEE Robotics and Automation Magazine*, **5**(1), (1998).

[3] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki, 'Multi-robot cooperation through incremental plan-merging', in *IEEE ICRA*, (1995).

[4] E.H. Durfee and V. Lesser, 'Partial global planning: A coordination framework for distributed hypothesis formation', *IEEE Transactions on Systems, Man and Cybernetics*, **21**(5), (1991).

[5] E.H. Durfee and T. A. Montgomery, 'Coordination as distributed search in a hierarchical behavior spac', *IEEE Transactions on Systems, Man and Cybernetics*, **21**(6), (1991).

[6] E. Ephrati, M. Perry, and J.S. Rosenschein, 'Plan execution motivation in multi-agent systems', in *AIPS*, (1994).

[7] G. Ferguson and J.F. Allen, 'Arguing about plans: plan representation and reasoning for mixed-initiative planning', in *AIPS*, (1994).

[8] S. Fleury, M. Herrb, and R. Chatila, 'Design of a modular architecture for autonomous robot', in *IEEE ICRA*, (1994).

[9] F. F. Ingrand, R. Chatila, and R. Alami F. Robert, 'Prs: A high level supervision and control language for autonomous mobile robots', in *IEEE ICRA*, (1996).

[10] N.R. Jennings, 'Controlling cooperative problem solving in industrial multi-agent systems using joint intention', *Artificial Intelligence*, **73**, (1995).

[11] C. Le Pape, 'A combination of centralized and distributed methods for multi-agent planning and scheduling', in *IEEE ICRA*, (1990).

[12] L.E. Parker, 'Heterogeneous multi-robot cooperation', Technical Report AITR-1465, MIT, (1994).

[13] S. Qutub, R. Alami, and F. Ingrand, 'How to Solve Deadlock Situations within the Plan-Merging Paradigm for Multi-robot Cooperation', in *IEEE IROS*, (1997).

[14] J.S. Rosenschein and G. Zlotkin, 'Designing conventions for automated negotiation', *AI Magazine*, **15**, (1994).

[15] Y. Shoham and M. Tennenholtz, 'On social laws for artificial societies: Off-line design', *Artificial Intelligence*, **734**, (1995).

[16] R.G. Smith, 'The contract net protocol: High-level communication and control in a distributed problem solver', *IEEE Transactions on Computers*, **C-29**(12), (1994).

[17] S. Yuta and S.Premvuti, 'Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots', in *IEEE IROS*, (1992).