ICOT Technical Report: TR-378

TR-378

Cooperative Problem Solving Approach for Portfolio Selection

by
H. Satoh, H. Ichiki and F. Suenaga(Fujitsu)

May, 1988

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Cooperative Problem Solving Approach for Portfolio Selection

Hideki Sato, Hiroki Iciki, and Fumiyo Suenaga

FUJITSU LIMITED

1015 Kamikodanaka, Nakahara-ku, Kawasaki 211, Japan

Abstract

This paper discusses cooperative problem solving as an approach to portfolio problems. Two groups of cooperative problem solvers are developed to select benchmark and management portfolios. In benchmark portfolio selection, an object centered architecture is used to organize a group of agents corresponding to each financial asset. Problem solving is controlled in fully distributed fashion. Based on information of other agents, each agent attempts to change its fund distribution in collaboration with other agents in parallel.

In management portfolio selection, a technique centered architecture is used to organize a group of agents corresponding to each management technique. Problem solving is controlled in semi-distributed fashion, enabling several management techniques to be fused to select quality portfolios. A single agent acts to propose improved portfolios, while other agents estimate them. Estimations of agents' goal attainment are exchanged to decide whether an improved portfolio is accepted from the standpoint of all agents. Also, opposing reasons for an improved portfolio are exchanged and used to reduce the search space to select alternatives.

To implement both groups of cooperative problem solvers, a parallel object oriented language, POOL, is developed on top of a parallel logic programming language, GHC. POOL provides content-based message passing and access control of slots shared by several processes as its unique features.

1. Introduction

Artificial Intelligence has been used to fabricate powerful problem solvers for

inherently complex applications such as medicine and geology. Financing has emerged as a challenging domain for artificial intelligence research and applications. Much attention has been focused upon developing financial expert systems to support professional decision making[1]. The portfolio problem, with the major application, has been increasing in importance with the today's ongoing liberalization of interest rates and internationalization of financing. Artificial intelligence is expected to provide effective computer support.

Portfolio problems deal with the selection of combinatorial investments to distribute fund efficiently and productively among financial assets such as stocks, bonds, the call market, and repurchase agreements. The investment framework[2] consists of three stages:
(1) Investment goals setup (PLAN): The characteristics of a fund are clarified and return and liquidity criteria, risk tolerance, and the like are set up. The benchmark portfolio is selected based on these considerations.
(2) Management strategy and tactics (DO): A management strategy is selected to increase performance by applying active and/or passive management techniques. The management portfolio is selected using the strategy.
(3) Evaluation of performance (SEE): Evaluation criteria are clarified and factor analysis is done for portfolio performance.

H. M. Markowitz[3] first formulated portfolio problems mathematically. Mathematical programming techniques have been conventionally used to apply both his work and its extensions in management science to select optimal benchmark portfolios. These techniques do not adequately model complex constraints and qualitative aspects using numeric formulae, however. Portfolio problems are non-linear[4]; there are many complex interactions among related goals, making models difficult to construct, modify, verify, and validate. We need a natural framework for selecting reasonable benchmark portfolios which take into consideration the qualitative aspects and modification used later for selecting management portfolios.

Host attempts to develop computer-support systems for selecting management portfolios by applying a single specific management technique. Experts attempt to apply techniques as many as needed to select quality management portfolios. The combination of

management techniques is non-sequential[4], in that the sequence in which they are applied to select a portfolio is not significant. This sequence is difficult to decide statically, and portfolios cannot be properly selected only by providing more than one techniques. A framework is needed to fuse management techniques. This framework must be able to combine new techniques with existing techniques easily. Modularity is also needed to program management techniques without involving complex interactions with other techniques in the system.

In cooperative problem solving[5], groups of intelligent agents attempt to solve problems. This conceivably could provide a basic framework for coping with the above problems. The technique provides high quality, high performance, expandability, and modularity. It still requires a great deal of work to put cooperative problem solving into actual use. We have been studying the application of cooperative problem solving to selecting benchmark and management portfolios. We developed a cooperative portfolio selection expert system in which multiple agents cooperate to select portfolios, using a parallel object oriented language, POOL, implemented on top of a parallel logic programming language, GHC.

The system overview is given in Section 2. GHC and POOL are explained in Section 3. We will then discuss frameworks of cooperative problem solving for selecting benchmark portfolios(Section 4) and management portfolios(Section 5). We close with conclusions.

2. System Overview

Figure 1 outlines portfolios selection. In selecting benchmark portfolios, users provide the system with the total amount of funds, financial assets, risk tolerance, term, and so on, as a plan specification. Agents corresponding to each financial asset are generated and assigned an initial allocation of funds to financial assets. A benchmark portfolio is selected through agent cooperation. The user evaluates the benchmark portfolio provided and can change plan specifications, if necessary, to select other benchmark portfolios.

In selecting management portfolios, either a benchmark portfolio or a management portfolio is provided as an initial portfolio, as are the management techniques to be used, trends in interest rates, goals of return, risk, liquidity, and so on. Agents corresponding to

management techniques are generated and the management portfolio is selected through agent cooperation. The user participates if necessary. The user is able to select a quality portfolio. In addition to selecting benchmark portfolios, the user evaluates the management portfolio provided and can change plan specifications, if necessary, to select other management portfolios.

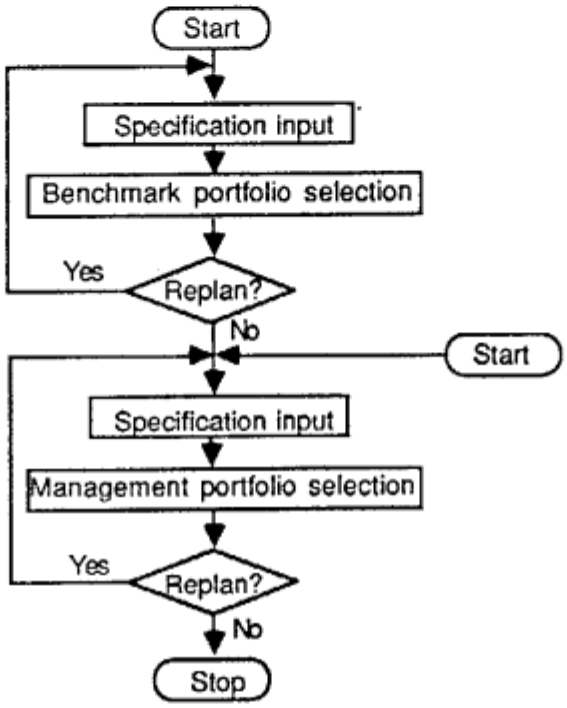


Figure 1 Overview of portfolio selection

3. Parallel Object Oriented Language: POOL

GHC[6] is used as a base language to develop cooperative problem solvers. GHC is based on first-order predicate logic as is Prolog[7] and a parallel logic programming language whose most important characteristic is parallel execution of goals. Unification in GHC is extended to include communication and synchronization between goals. In the guard of a clause, variables in the caller goal can be read, but attempting to instantiate the variable suspends the goal. The suspended goal is resumed when another goal instantiates the variable. There are no restrictions on the unification of the body of the clause.

Although GHC can deal concisely with communication and synchronization in parallel

programming, it is too primitive to develop large-scale parallel application programs. To describe such programs more simply, concisely, and directly requires the introduction of a highly abstract concept. We chose object oriented paradigm[9] for this purpose and designed POOL on top of GHC in the framework of a perpetual process using stream communication[8].

Programs in POOL are a collection of classes that define objects and are not themselves objects. An object has slots for storing data related to itself and methods to specify its behavior in the form of the guarded Horn clause. The superclass-subclass relationship can be defined between any two classes, except where they form loops by chaining relationships. A class may inherit slot and method description from more than one superclasses. Built-in primitives are used for slot access, message passing, and the generation and deletion of objects.

3.1 Content-Based Message Passing

Message passing in POOL is basically treated as an asynchronous communication. That is, a message sender can continue executing its process independently of the processes of message receivers. Synchronization occurs only where the sender must refer to values computed by the receivers and the values are not returned from the receivers at a reference time. This synchronization is done by GHC.

To develop cooperative problem solvers, content-based message passing is needed in addition to the one-to-one message passing between any two objects in the conventional object oriented language[9]. In POOL, the capability of receiver designation based on function (similar to the audience restriction in [10], [11]) and conditional restriction (similar to the eligibility criteria in [12]) are included with broadcasting. Receivers are specified by the function and/or restriction conditions they must satisfy as well as by object identifiers. In *Fig. 2*, an agent sends a message to a group of agents whose function is **short_term_bond** and that satisfy the **region(...)** condition. This reduces unnecessary message passing, and enhances the expandability of cooperative problem solving, the descriptiveness, readability of programs in POOL.

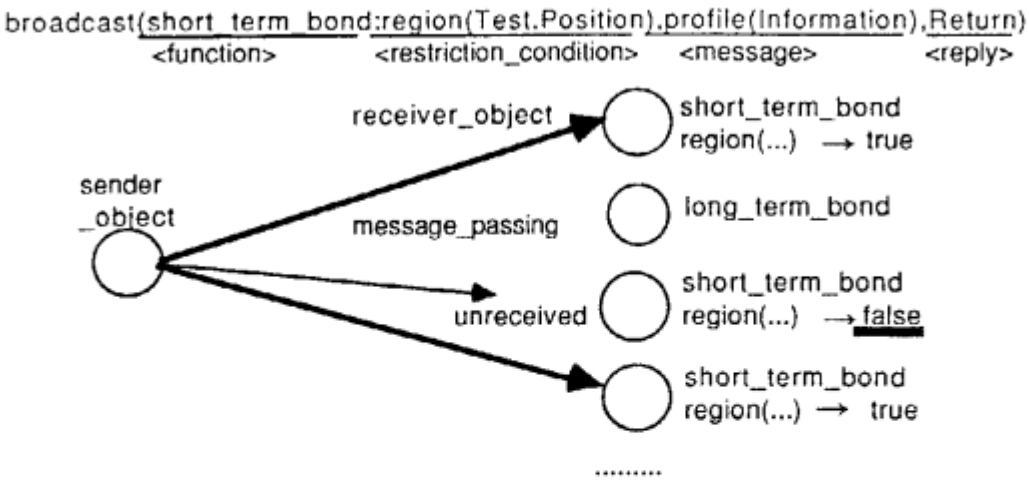


Figure 2 Content-based message passing

3.2 Access Control of Shared Slots

Access control is needed to keep shared data consistent in parallel processing. Data stored in object slots is shared by several processes in POOL. To prevent accesses to a same slot by several simultaneous processes from making data inconsistent, the **read** and **write** access primitives are nondecomposable, but this does not guarantee consistency if processes try to **read** access, to compute new values, and to **write** to shared slots(*Fig. 3*). A **read-&-write** access primitive is added to read the value of a slot and to set the variable which can be updated in the slot at the time. Using the primitive, a process which reads variable set as the value of a slot is suspended by GHC synchronization until another process assigns a updated value to the variable(*Fig.4*). This guarantees the consistency of the individual slot because execution of the processes accessing shared slots is serialized.

The **read-&-write** access primitive guarantees single-slot consistency, but that for several related slots is another problem. Merely issuing several **read-&-write** access primitives may cause processes to deadlock. One way of avoiding this might be to have processes issue **read-&-write** access primitives in a pre-defined sequence, e.g., lexicographic sequencing of slot names. This does not work here, however, because the issuing of **read-&-write** access primitives is by executing corresponding goals in GHC in parallel, which makes pre-defined sequencing meaningless. We solved the problem by adding a **multiple-read-&-write** access primitive to read and write to several slots at

once, enabling processes to read the consistent state and to change to a new consistent state. This primitive keeps processes deadlock-free, because several slots are occupied together and the waiting relation between processes never loops.

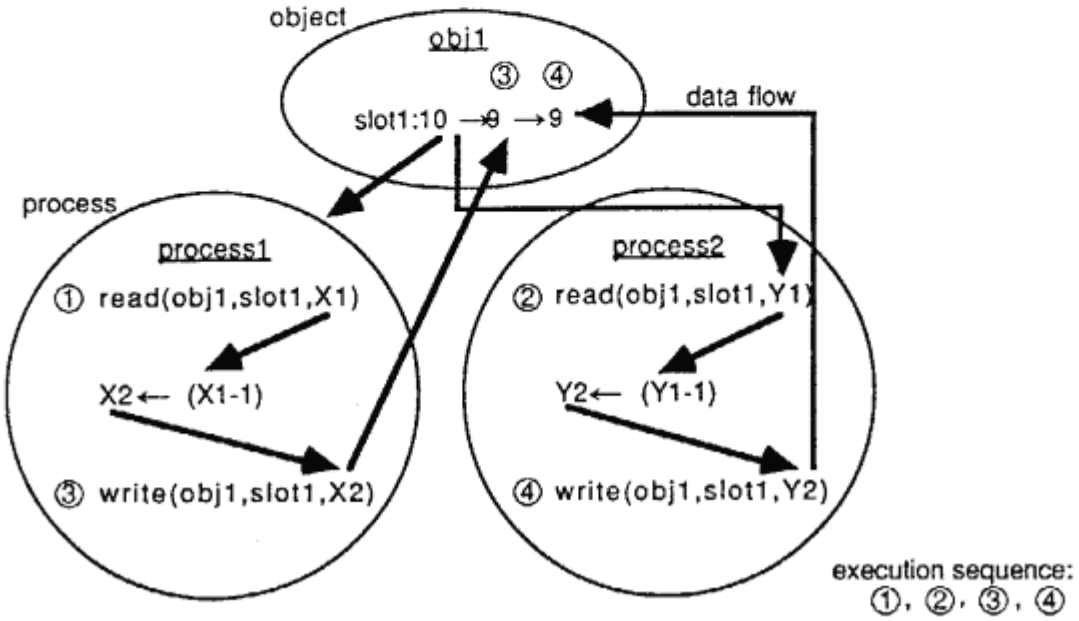


Figure 3 Inconsistent data flow to a slot

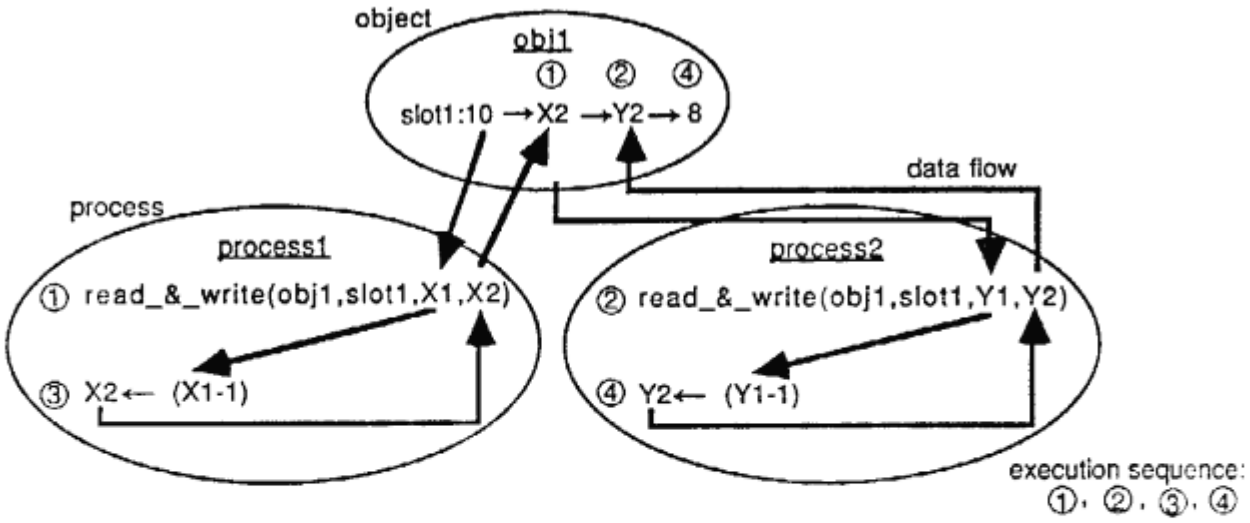


Figure 4 Consistent data flow to a slot

4. Benchmark Portfolio Selection

We propose an object centered architecture[13] for cooperative problem solving to select benchmark portfolios, based on observation of the distribution of financial assets, each of which has a corresponding agents in the architecture. Given the architecture, knowledge on object relations and interactions, qualitative aspects, and heuristics are organized in a conceptually natural form.

The world consisting of the return and risk coordinates axes is used to select benchmark portfolios. Each agent corresponds to a financial asset and has a position specified by the return and risk of the financial asset(*Fig. 5*). Agents are active, attempting to change their fund distribution in collaboration with other agents to make them effective in a benchmark portfolio.

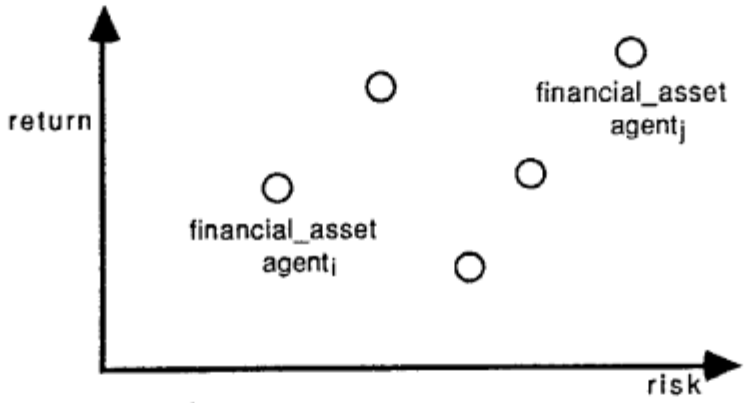


Figure 5 Financial_asset agents in the return_risk world

4.1 Problem Solving by Agents

The selection of benchmark portfolios is controlled in fully distributed fashion. There are no master-slave relationships among agents in problem solving. An agent attempts to repeat changing its fund distribution in collaboration with other agents by performing such activities as gathering and maintaining information of other agents in the world, selecting a partner out of "agents of concern", a set of agents having probability to improve its performance by combining with it, and negotiating a new fund distribution.

Gathering and maintaining information of other agents in the world

Each agent keeps world information that includes information on other "agents of

concern". Information is initially gathered by requests to other agents, then maintained by notification triggers from other agents reporting changes in the world state. For an agent, not all other agents are "of concern". The number of other agents is reduced by using heuristics to restrict the scope of "agents of concern", for example, by the positions of other agents relative to an agent(*Fig. 6*). With regard to return and risk, agents in quadrants 1 and 3 relative to the position of an agent tend to supplement each other with the agent, while agents in quadrants 2 and 4 tend to conflict with the agent.

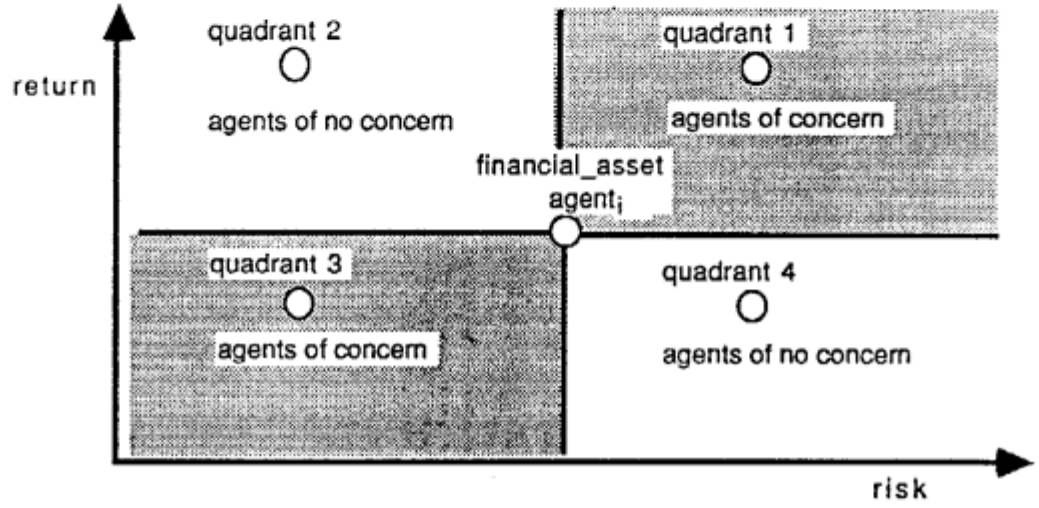


Figure 6 Hueristic to restrict "agents of concern"

Selecting a partner out of "agents of concern"

An agent selects a partner from of "agents of concern", with which it can improve its performance of return, risk, and other parameters among "agents of concern" by changing their fund distribution. Each agent is able to evaluate other agents, and how to evaluate is shared among all agents.

Negotiating a new fund distribution

An agent negotiates with a selected agent fund redistribution. The conclusion of agreement to select each other as partners is imposed on all agents to ensure globally coherent behavior. Two agents form a virtual composite agent that takes a position corresponding to new values of return and risk[2] in the world(*Fig. 7*), and attempts to

repeat the above activities like a single agent. When activities of all agents can no longer be repeated, that is, there are no possible agreements, a benchmark portfolio is selected as the combination of their fund distribution.

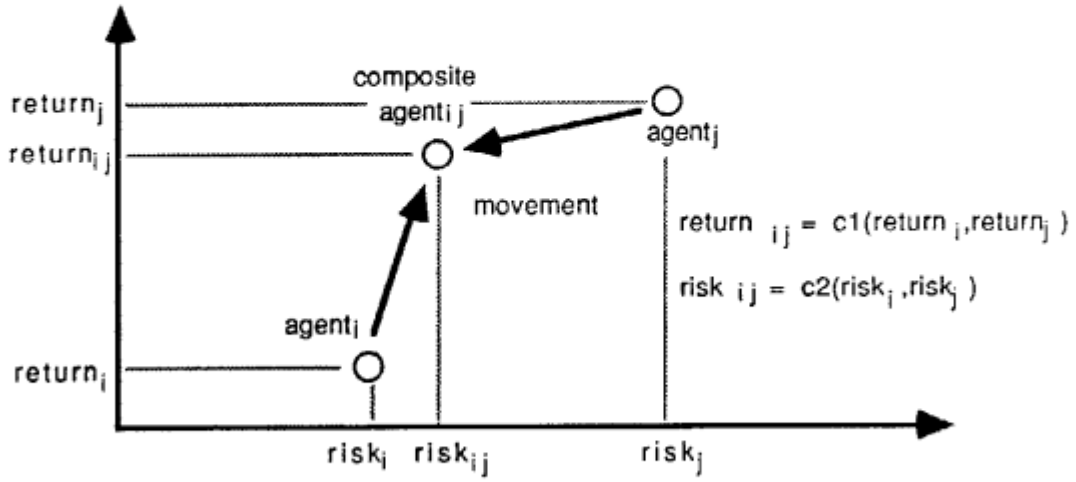


Figure 7 Financial_asset agent

4.2 Asynchronous Event Manager

Agents change their fund distribution based on their own world information in parallel. Two agents agreeing to change of their distribution form a composite agent to perform activities later as a one and other agents concerned with the moving agents must update their world information. Updating starts a new round of activities in line with the new world state. These events result from parallel agent activities and occur asynchronously. The control of agent activities requires that the occurrence of events be monitored and that needed processes be invoked.

An agent changing the world state sends a message to notify directly other agents having concern toward it(*Fig. 8*). Each agent has an asynchronous event manager to manage asynchronous events relating to it. After receiving a message notifying a change, process of the agent attempts to update its world information and to inform the manager that an event occurred. The manager has event-task association knowledge and retrieves tasks relating to the event, then invokes task processes.

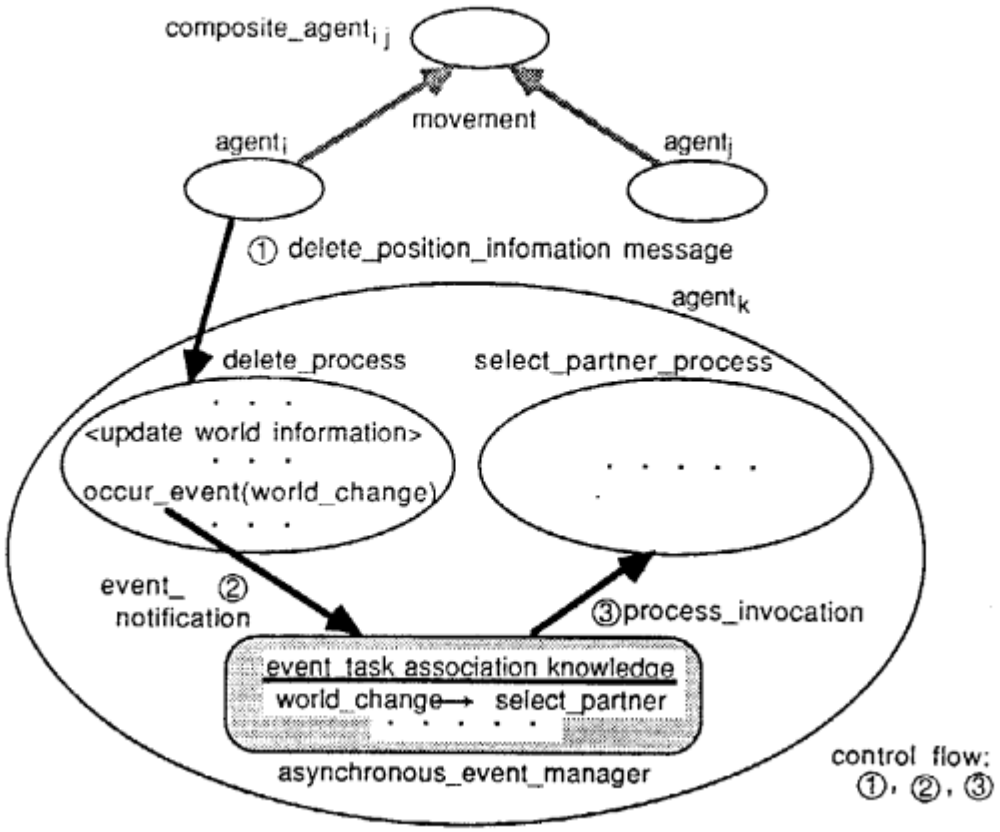


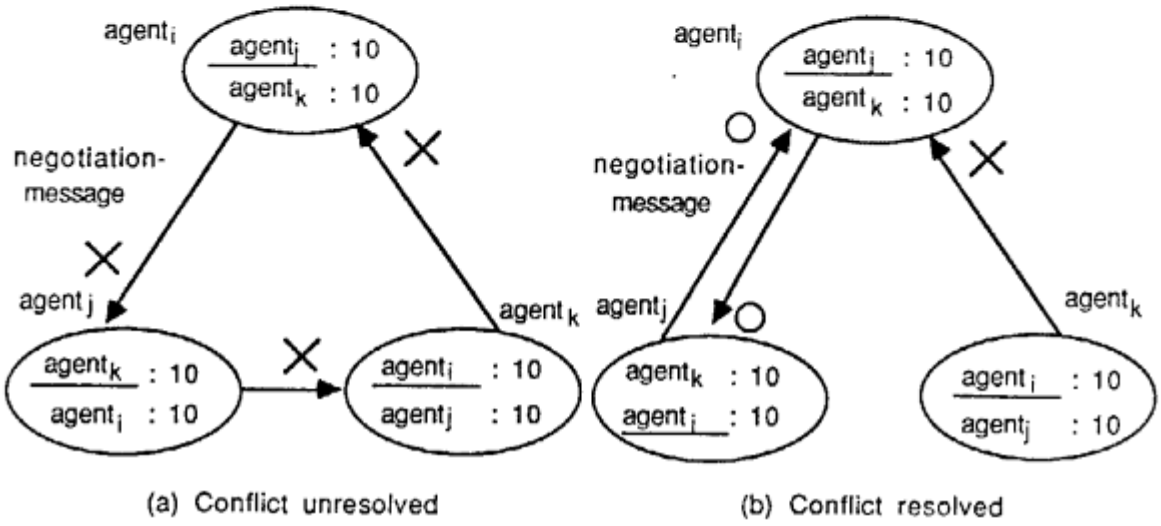
Figure 8 Asynchronous event manager

Event management is distributed among asynchronous event managers serving corresponding agents. In contrast to global centralized management, managers can prevent bottlenecks and improve overall reliability. Introducing managers treats relationships among processes indirectly, not directly, if a process detecting events invokes processes. This makes managing processes flexible for agents.

4.3 Conflict Resolution

An agent selects a partner estimated to be the highest among "agents of concern" and sends a message to negotiate fund distribution. If the second agent accepts the first agent as a partner, their negotiation is agreed. Conflicts occur if more than one agents is considered to be the highest "of concern". Conflicting agents forming a conflict loop by chaining sender-receiver relationships, the conflict cannot be resolved, even if they are potentially able to reach some agreement(*Fig. 9*(a)).

A simple but effective way to resolve conflicts is to lexicographically sequence agents' names. Each agent selects the agent that is at the top in the lexicographic sequence of conflicting agents' names. Using the mean, the top agent in the lexicographic sequence among agents in a potential conflict loop selects the second top agent as its partner, and remaining agents select top agent as their partners. The mean results in agreement between the top and the second top agents, and no conflict loop forms(*Fig. 9*(b)).



(a) Conflict unresolved (b) Conflict resolved

Figure 9 Conflict resolution

5. Management Portfolio Selection

A management portfolio is selected by applying management techniques to a portfolio whose initial value is either a benchmark portfolio or a management portfolio. A technique centered architecture is used to select management portfolios, based on the observation of the distribution of management techniques, each of which corresponds to individual agents in the architecture. An agent must have two functions:

(1) It has goals of its own and is able to select a portfolio improved from the standpoint considering the intentions of other agents. Its goals have ranges of acceptance and it decides level of goal flexibly to meet situations in global problem solving.

(2) It can estimate improved portfolios selected by other agents as well as its own. It is also able to return reason, if it does not agree to an improved portfolio selected by other agents.

5.1 Problem Solving Outline

Selection of management portfolios is controlled in semi-distributed fashion. There are no master-slave relationships. A single agent acts to propose improved portfolios. Other agents estimate proposed portfolios. The right to improve is assigned depending on the problem solving situation. Portfolio problems are non-linear making interactions among several management techniques complex and resulting in semi-distributed control that enables portfolio improvements gradually by the participation of all agents.

The agent with the right to improve selects an improved portfolio as follows(*Fig. 10*): The agent first decides the level of the goal for improvement based on its level and/or other agents' levels of goal attainment. It then selects a candidate portfolio satisfying the goal's level. Other agents assess the candidate and give the agent proposing the portfolio their estimations and decisions of acceptance. These are used to decide whether the portfolio is to be accepted or rejected by the group of agents. When it is accepted, the candidate has a new portfolio and the right to improve goes to the next agent. If it is rejected and alternatives exist, the same procedure is repeated. Otherwise, the right to improve goes to the next agent with no change of portfolio.

Table 1 Evaluation of a possible improved portfolio

Level of goal attainment	Estimation
Attainment of the maximum goal level	Best
Attainment of reasonable goal level between the maximum and the minimum. They are divided into several levels as the agent divides.	Good
Nonattainment of the minimum goal level	Bad

Table 2 Decision on acceptance concerning a improved portfolio

Change in level of goal attainment	Opinion
Ascent in level of goal attainment from an agent's position	Approval
No change in level of goal attainment from an agent's own position	Abstention
Descent in level of goal from an agent's own position	Opposition

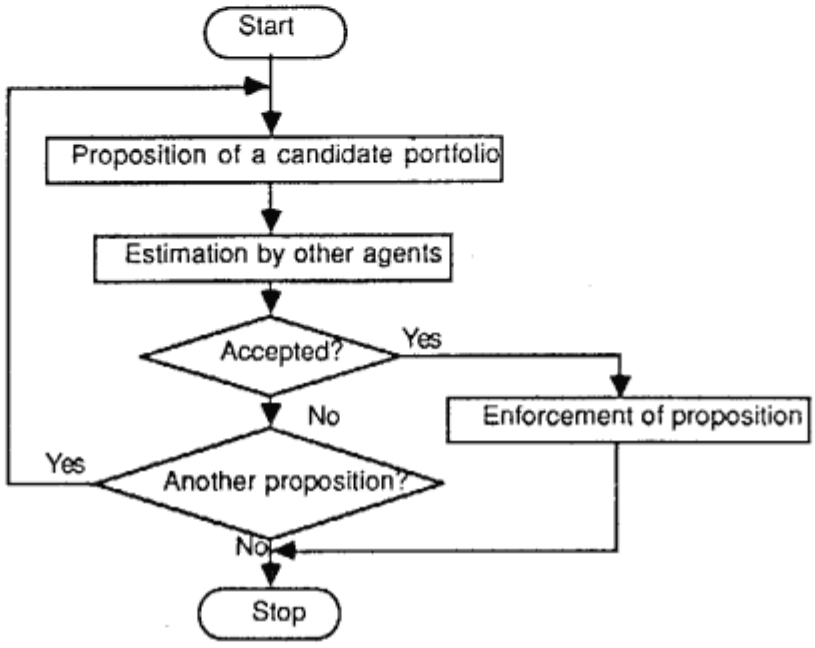


Figure 10 General procedure followed by an agent with the right of improvement

The portfolio selected by a group of agents must satisfy at least the worst-case levels of all agents' goals. Each agent uses its own strategy for achieving its goals. An agent evaluates a portfolio as outlined in *Table 1*. The acceptance of an improved portfolio by an agent is decided taking into account the change between a portfolio and an improved one as outlined in *Table 2*.

5.2 Strategies in Problem Solving

We use three problem solving strategies; remedial, relaxation, and competition(*Table 3*). Both the remedial and relaxation strategies are used to achieve the condition required for a portfolio to be selected; it must satisfy at least the worst-case levels of all agents'

goals. Remedial strategy is used by agents at the bad level of goal attainment. Relaxation strategy is used by agents at the best level. Competition strategy is used to raise goals toward the best level by agents at the good level.

Table 3 Problem solving strategies

Strategy	Remedial	Relaxation	Competition
Purpose	To help agents at the bad level of goal attainment	To help agents at the bad level of goal attainment by making agents at the best level relax	To make agents at good level of goal attainment compete toward the best level
Agent with the right to improve	agent at the bad level of goal attainment	agent at the best level of goal attainment	agent at the good level of goal attainment
Condition to accept an improved portfolio	Decrease in the number of agents at the bad level of goal attainment	Decrease in the number of agents at the bad level of goal attainment	No agents at the bad level of goal attainment, and either acqusition of majority consensus or no oppositions
Next agent to be assigned the right to improve	Agent at the bad level of goal attainment which was assigned the right to improve least recently	Agent at the best level of goal attainment which was assigned the right to improve least recently	Agent at the good level of goal attainment which was assigned the right to improve least recently
Termination condition	No agents at the bad level of goal atatinment, or impossible to improve more further	No agents at the bad level of goal attainment, or impossible to improve more further	impossible to improve more further

Whether an improved portfolio is accepted is decided by evaluation(*Table 1*) in remedial and relaxation strategies. In competition strategy, both evaluation and a decision of acceptance by an agent(*Table 2*) are used for the decision of acceptance from the standpoint of all the agents. When there are no agents at the bad level of goal attainment for an improved portfolio under the strategy, decision of acceptance by all agents is based on the principle of majority consensus or one of no oppositions which are often used in the society

of human beings. In particular, the latter principle guarantees Pareto optimality[14].

Use of a strategy ends when there are no agents qualified for improvement. Also, use of remedial and relaxation strategies ends when there are no agents at the bad levels of goal attainment. *Fig. 11* shows the transition among strategies. If there are agents at the bad levels of goal attainment when use of remedial and relaxation strategies ends, a group of agents fails to select a portfolio. When use of competition strategy ends, a group of agents selects a portfolio successfully.

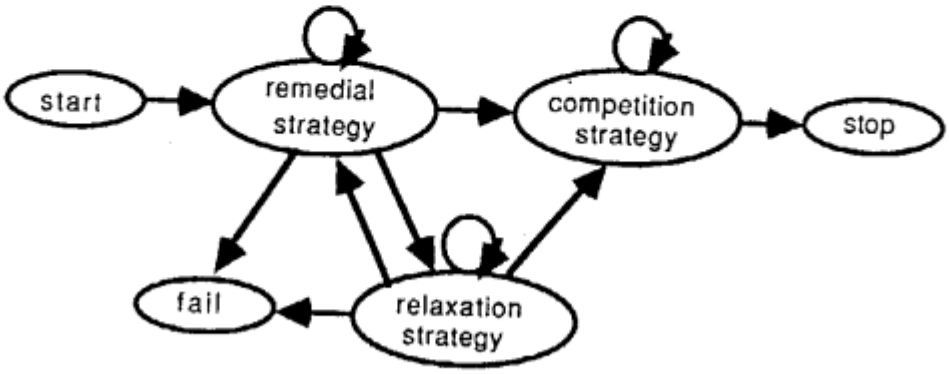


Figure 11 Transition among problem solving strategies

5.3 An Example of Improvement

An example of improvement by an agent with the right to improve is presented in this subsection. To simplify a problem, three agents(*Table 4*) and two financial assets(*Table 5*) are used. Each agent has his own goal having range of acceptance. For example, the goal of D-agent is " Capital gain 2.2% at least the worst-case, and 2.7% if possible". The details of management techniques about portfolios, knowledge on bonds and so on are not explained in this paper but textbooks(e.g., [2]) should be referred for them.

Let fund distribution in portfolio be 48% for A-bond and 52% for B-bond(Notation <48,52> is used in the followings). Also, let an agent with the right to improve be D-agent. Estimations from the standpoint of agents' goals are 2.32% for D-agent, 14.36% for P-agent, and 4% for B-agent. Accordingly, goal attainments of all agents are at the good level. Therefore, competition problem solving strategy is used. Whether an improved portfolio is accepted is based on the principle of majority consensus. The unit of fund distribution is 1%.

Table 4 Profile of agents

Agent	D-agent	P-agent	B-agent
Overview	Agent whose technique is based on difference of yield for residual term	Agent whose technique is based on predicion of interest rate	Agent who manages assignment of fund to short term and long term bonds
Goal	Capital gain 2.2% at least the worst-case and2.7% if possible	Holding period return 13.5% at least the worst-case and 15.0% if possible	Difference of fund assignment 10% and 0% if possible

Table 5 Profile of bonds

Bond	A-bond	B-bond
Kind	Short term bond	Long term bond
Maturity	2 years	9 years
Coupon	8%	8%
Yield & price at present	6.0%, 103.6yen	11.0%, 86.4yen
Yield & price after half a year, & holding period return based on prediction of interest rate	Yield 5.0% Price 104.2yen Holding period return 19.4%	Yield 10.0% Price 90.8yen Holding period return 8.9%
Yield & price of bond with fewer maturity by half a year, & capital gain based on yield cureve	Bond with maturity 1.5 years Yield 4.4% Price 105.1yen Capital gain 2.9%	Bond with maturity 8.5 years Yield 10.9% Price 87.2yen Capital gain 1.8%

Range of acceptance an agent's goal has is divided several sub-ranges in its own way. Given portfolio <48,52> whose capital gain is 2.32%, D-agent decides to raise capital gain in the sub-range between 2.34% and 2.40%, which is better by one rank than the sub-range including capital gain 2.32%. To attain this, D-agent sets up constraint that fund distribution of A-bond is over 48%, because capital gain is a weighted average of all bonds and capital gain of A-bond is higher than that of B-bond(*Table 5*). D-agent selects a

candidate portfolio <55,45> whose capital gain is 2.40% and asks both P-agent and B-agent to evaluate it. Both P-agent and B-agent return to D-agent their opposing intentions and reasons that fund distribution of A-bond is too high. With their opposing reasons, D-agent changes constraint into new one that fund distribution of A-bond is over 48%, and less than 55%. This reduces the search space to select alternatives. Fund distribution of A-bond, 51%, 52%, 53%, and 54% satisfy the desired sub-range and the constraint. D-agent selects 52% as fund distribution of A-bond using heuristic that middle value of range restricted by constraint is accepted more probably than value near ends of range. Then, D-agent asks both P-agent and B-agent to evaluate an improved portfolio <52,48> again. P-agent opposes again for the same reason as before and B-agent returns intention of abstention. With P-agent's reason, D-agent changes constraint into new one that fund distribution of A-bond is over 48%, and less than 52%. Then, D-agent selects a candidate portfolio <51,49> and asks both P-agent and B-agent to evaluate it. P-agent returns approval and B-agent returns opposition as before. This time, D-agent successes to get majority consensus including his own approval and the portfolio <51,49> is accepted by all agents.

6. Conclusions

This paper discusses cooperative problem solving frameworks for portfolio problems such as the selection of benchmark and management portfolios. We developed a parallel object oriented language, POOL, on top of a parallel logic programming language, GHC, to fabricate two groups of cooperative problem solvers for portfolio problems. GHC provides communication and synchronization based on unification, which POOL makes use of to the utmost. Unique POOL features are its content-based message passing and access control of shared slots. Content-based message passing is effective in making cooperative systems descriptive and expandable. Access control of shared slots is made possible by using GHC synchronization mechanism.

To select benchmark portfolios, an object centered architecture is used to organize agents corresponding to financial assets. Problem solving is controlled in fully distributed

fashion for agents to change their fund distribution among other agents. These agents have clues to include qualitative information and/or heuristics to prevent combinatorial explosion. An asynchronous event manager for each agent controls processes related to the occurrence of external asynchronous events as local. Conflicts during changing fund distribution are resolved by using a lexicographical sequence of agents' names and this is commonly shared among all agents.

To select management portfolios, a techniques centered architecture is used to organize agents corresponding to management techniques. Problem solving is controlled in semi-distributed fashion, because portfolio problems are non-linear. Several management techniques are fused to select quality management portfolios. A single agent acts to propose improved portfolios, while other agents estimate them. Estimation of agents' goal attainment are exchanged to decide whether an improved portfolio is accepted from the standpoint of all agents. Opposing reasons for an improved portfolio are also exchanged and used to reduce the search space to select alternatives. Evaluation, decisions on the acceptance of an improved portfolio, and problem solving strategies are designed based on human-like action. The expandability needed to implement new management techniques is guaranteed because a user can freely specify agents participating in problem solving to select management portfolios. Modularity in defining each agent independently of other agents is effective, especially in developing large-scale and/or complex systems.

Acknowledgments

This work is part of the activities undertaken in the Fifth Generation Computer Systems (FGCS) Project of Japan. We thank Mr. Fujii, manager of the Fifth Section of the Institute for New Generation Computer Technology (ICOT) for his encouragement and support. We also thank Mr. Hayashi, manager of the Artificial Intelligence Laboratort of Fujitsu Laboratories Ltd., and Mr. Yamamoto, manager of the First Section, of the Artificial Intelligence Laboratory of Fujitsu Laboratories Ltd., for their guidance.

References

[1] Apte, C. and Kastner, J. (ed.), "Special issues on Financial Applications", IEEE EXPERT,

vol.2, no.3, Fall, 1987

[2] Nomura Research Institute (ed.), "Bond Management and Investment Strategy", Monetary and Financial Situation Study Aggregate Corporation, July, 1981, (in Japanese)

[3] Markowitz, H. M., "Portfolio Selection: Efficient Diversification of Investments", Wiley, New York, 1959

[4] Clemons, E. K., "A Decision Support System Architecture for subjective,loosely constrained, data-intensive problem domains", WP 84-05-05, The Department of Decision Sciences, University of Pennsylvania, 1986

[5] Smith, R. G., "Report on the 1984 Distributed Artificial Intelligence", AI Magazine, Fall, 1985

[6] Ueda, K., "Guarded Horn Clauses", Technical Report TR-103, ICOT, 1985

[7] Bowen, D. L. et al. (ed.), "DEC System-10 Prolog User's Manual", The Department of Artificial Intelligence, University of Edinburgh, 1982

[8] Shapiro, E. and Takeuchi, A., "Object Oriented Programming in Concurrent Programming", New Generation Computing, 1(1983), Ohmsha Ltd. and Springer-Verlag

[9] Goldberg, A. and Robson, D., "Smalltalk-80: The Language and Its Implementation", Addison-Wesly, Reading, 1983

[10] Parunak, H. V. D., "Manufacturing Experience with the Contract Net", in Proceedings of 1985 Distributed Artificial Intelligence Workshop, pp.67-91, December, 1985

[11] Parunak, H. V. D. et al., "Fractal Actors for Distributed Manufacturing Control", in Proceedings of 2nd IEEE Conference of AI Applications, December, 1985, pp.653-660

[12] Smith, R. G. and Davis, R., "Frameworks for Cooperation in Distributed Problem Solving", IEEE Transactions on Syst., Man., Cybern., vol.SMC-11, pp.61-70, January, 1981

[13] Thorndyke, P. W., McArthur, D., and Cammarata, S., "Autopilot: A Distributed Planner for Air Fleet Control", in Proceedings of 7th Int. Joint Conf. Artificial Intelligence, Vancouver, Canada, August, 1981, pp.171-177

[14] Saeki, Y., "Theory of Decision: An Introduction to Social Decision Theory", University of Tokyo Press, April, 1980, (in Japanese)