Obtaining an Optimal MAS Configuration for Agent-Enhanced Mining Using Constraint Optimization

Chayapol Moemeng, Can Wang, and Longbing Cao

Quantum Computing and Intelligent Systems,
Faulty of Engineering and Information Technology,
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia
{mchayapol,cawang,lbcao}@it.uts.edu.au

Abstract. We investigate an interaction mechanism between agents and data mining, and focus on agent-enhanced mining. Existing data mining tools use workflow to capture user requirements. The workflow enactment can be improved with a suitable underlying execution layer, which is a Multi-Agent System (MAS). From this perspective, we propose a strategy to obtain an optimal MAS configuration from a given workflow when resource access restrictions and communication cost constraints are concerned, which is essentially a constraint optimization problem. In this paper, we show how workflow is modeled in the way that can be optimized, and how the optimized model is used to obtain an optimal MAS configuration. Finally, we demonstrate that our strategy can improve the load balancing and reduce the communication cost during the workflow enactment.

Keywords: Constraint Optimization, Workflow Management System, Agent and Data Mining Interaction.

1 Introduction

A challenge in agent and data mining interaction (ADMI) is the interaction itself: whether agent enhances the data mining process, or data mining is used to improve agent intelligence [3]. Given the current pace of research progression from these two fields, introducing an entirely new mechanism would face the issues in terms of acceptance and adoption by the ADMI community.

At first, let us look at the successful existing data analysis and mining tools (data mining tools, for short). Take SAS Analytics[1], RapidMiner[2], KNIME[3], as examples, they manage complicated data analysis and mining components

[1] SAS website: http://www.sas.com/
[2] Rapid-i website: http://rapid-i.com/
[3] KNIME website: http://www.knime.org

L. Cao et al. (Eds.): ADMI 2011, LNAI 7103, pp. 46–57, 2012.

through workflow-style tools. Workflow not only provides a high level of visualization which increases the usability of the system, but also eases system developments in terms of re-usability from the computational resources, such as data analysis and mining components. As a result, users can define arbitrary workflows for their requirements with the support of these application tools. As for today, Workflow Management System (WfMS) has become a de facto standard for data mining tools [9]. Contemporarily, major efforts on improving WfMS performance by using different system architectures and engineering strategies [1,14] have been made.

Besides, in terms of data mining techniques, there exist two major constraints which significantly impact the performance of the data mining process: *large data sets* and *resource access restriction* [8]. In fact, transmitting a large amount of data over the network could degrade such performance due to the enforced resource access restriction, and may lead the system into inconsistency state and various costs [1] which involve message passing for workflow management activities. Consequently, handling the aforementioned constraints together with the performance improvement at the same time is a challenging task.

However, from the standpoint of our interests in ADMI, adjustable anatomy of Agent-based Workflow Management System (A-WfMS) has been investigated to support dynamic constraints analysis recently [2]. Within A-WfMS, essentially, the workflow functions as a process descriptor and a Multi-Agent System (MAS) handles the workflow engine by coordinating the workflow enactment [7,11,5,10].

Our recent work [9] has shown the successful use of workflow system as an interaction platform between agent and data mining. Further, in this paper, we focus on how to obtain the optimal MAS configuration for the agent-enhanced data mining when given an optimized workflow model. In fact, this problem is a class of Constraints Optimization Problem (COP) in which the hard constraint (resource access restriction) must be satisfied and the soft constraint (data transfer minimization) can be optimized.

The main contributions of this paper are in threefold: (i) we explore a type of A-WfMS data mining process explicitly, (ii) we present a strategy to interact between agents and data mining in terms of agent-enhanced mining based on constraint optimization framework, and (iii) we provide the ADMI community with a mechanism that can be quickly and inexpensively accepted and adopted.

The rest of the paper is structured as follows. Section 2 reviews the related work for this paper. Problem statement, including preliminary assumptions, case description, and relevant definitions, is explained in Section 3 in which the POCL plan [13] is used to present our task description. Section 4 specifies the constraint optimization and the MAS configuration schemes individually. Afterwards, we evaluate the strategy and analyse the results in Section 5. Finally, we end the paper in Section 6.

2 Related Work

Integration of agents and WfMS have been introduced in the late 1990s. Pioneer work [7,11] argued that agents are suitable for workflows since the nature of

the requirements could always evolve over time. Therefore, automatic process improvement is desirable, which can then intelligently adapt to the changing environmental conditions. Recent A-WfMSs, JBees [5] and i-Analyst [10], use collaborating software agents as the basis of the systems provides more flexibility than existing WfMSs. JBees uses Coloured Petri nets as process formalism However, JBees' allocation of process agent (workflow engine) does not concern the cost of communication and resource access restrictions. On the other hand, i-Analyst is specifically designed for data mining process using agent technology for workflow execution layer. It provides a range of capabilities, including workflow construction, workflow management, algorithm development, resource management, workflow enactment, and reporting. The mechanism for MAS configuration of i-Analyst is described in this paper.

In later years, attentions have been moved to the implementation techniques. The increasing growth of the Internet has played a major role in the development of WfMS. Recently and remarkably, web Services is a promising method to provide the computational resources for workflow, and such workflow benefits from the support for distributed process models [12]. Although web services technology is becoming a main stream in workflow integration, Huhns [6] argued that web services alone may not completely fulfil distributed WfMS requirements due to the fact that web services know only about themselves, rather than any meta-level awareness; web services are not designed to utilize or understand ontologies; besides, web services are not capable of autonomous action, intentional communication, or deliberatively cooperative behavior. Accordingly, Buhler and Buhler [2] showed a critical survey of workflow, web services, and agent technologies to make a point that web services and MAS will become imperative parts of the future WfMS development. To our extend, we aim at an agent-enhanced data mining system in which computational resources (operations) are not required to be web services. The location requirement of the web service does not support our approach, because rather than transmitting large data over the network to the designated operation at a service provider, we prefer the operation to be transmitted to the data site instead.

Some works attempting to improve the performance of A-WfMS have become incrementally popular. For instance, Yoo et al. [14] proposed an architecture that involves agents in workflow engine layer, therefore, agents can help distribute workloads of a naming/location server and a workflow engine effectively. Bauer and Dadam [1] proposed a variable server assignment for distributed WfMS which allows dynamic server assignment without expensive run-time analysis. Their approaches use the advanced techniques in system engineering to help minimize the communication load in the distributed system, while our approach takes great advantage of the pre-construct configuration which has its own unique benefit: that is once the agent organization has been acquired and tasks have been allocated to the corresponding agents, they may perform tasks in advance, such as loading independent data and libraries, without having to wait for dependent task completions. In our understanding, improving the performance means minimizing the related costs in A-WfMS.

3 Problem Statement

This section depicts the building blocks of this paper. We propose, firstly, the preconditions of the involved data mining tool. Secondly, the case to be used in the rest of the paper is specified. And finally, the relevant definitions are formalized. In this work, the concerned system is a data mining tool with an embedded WfMS. The WfMS has the information of the size of data sources. The workflow execution layer is deployed on a group of computers (hosts) on a network. Each host has an agent platform to perform as a workflow engine. Lastly, the topology of the network does not change significantly. We have implemented such system in our previous work [10].

3.1 Case Description

For the rest of the paper, we set up a workflow and describe the MAS configuration to be obtained from the above one.

To begin with, Fig. 1 is a workflow that consists of six tasks. All the tasks are connected with each other by one or more directed arcs. Each arc is inscribed with a cause c_i and a cost in a numeric value. A cause c is both an effect of task t_i and a precondition of task t_j [13] which can be written as $t_i \xrightarrow{c} t_j$.

Highlighted tasks t_1 and t_4 are *location-specific tasks*, which must be executed at the specific location only; while other tasks are *non-location-specific*.

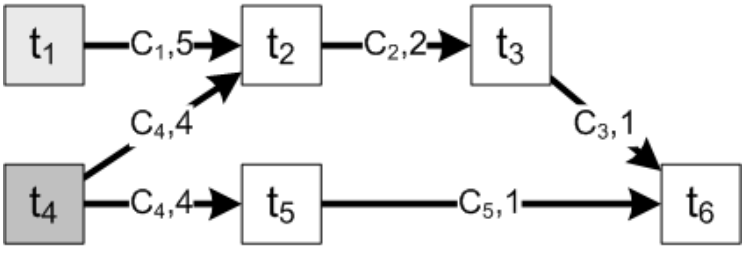

Fig. 1. A Workflow

In addition, MAS configuration is a document that describes the organization of agents in a platform. The description of the configuration is varied by the types of agent platform. Let us use a generic agent platform, WADE, for our common understanding. WADE is an extension of JADE (Java Agent Development Framework) [4]. Listing 1.1 shows a sample of MAS configuration used in WADE. Specifically, an agent *platform* is composed of multiple hosts. Each *host* may contain multiple agent containers, in which each *container* may have multiple agents. *Agents*, in particular, are type specific. In this example, agents are of *Workflow Engine (WE) Agent* type. Note that every element (host, container, and agent) must have a specific name.

[4] WADE website, http://jade.tilab.com/wade/index.html

```
<platform name="Configuration-1">
 <hosts>
  <host name="host1">
   <containers>
    <container name="Execution-Node-1">
     <agents>
      <agent name="performer1" type="WE Agent" />
      <agent name="performer2" type="WE Agent" />
     </agents>
    </container>
   </containers>
  </host>
 </hosts>
</platform>
```

Listing 1.1: Sample MAS Configuration

3.2 Definitions

We desire to obtain the MAS configuration that is optimal for a workflow. In order to get such document, we establish the following definitions. In this work, our definition of a multi-agent plan extends that of the Partial-Order Causal-Link (POCL) plan. The POCL plan definition has been well-established in the planning community [13]. Our definition is also inspired by the framework proposed by Cox and Durfee [4].

Definition 1. *A **Localized POCL (L-POCL) plan** is a tuple* $P = \langle T, \succ_T, \succ_C, L_T, \mathscr{C} \rangle$ *where*

- T *is a set of the tasks,*
- $\succ_T$ *is the temporal orders on* T*, where* $e \in \succ_T$ *is a tuple* $\langle t_i, t_j \rangle$ *with* $t_i, t_j \in T$*,*
- $\succ_C$ *is the causal partial orders on* T*, where* $e \in \succ_C$ *is a tuple* $\langle t_i, t_j, c \rangle$ *with* $t_i, t_j \in T$ *and* c *is a cause,*
- L_T *is a set of the locations and the corresponding tasks on* T*, where* $e \in L_T$ *is a tuple* $\langle l_i, t_j \rangle$ *with location* $l_i \in L$ *and task* $t_j \in T$*,*
- $\mathscr{C}$ *is the cost function mapping* $\mathscr{C} : T \times T \rightarrow \{0, \Re\}$*, this function* $\mathscr{C}(t_i, t_j) = 0$ *if* $l_{t_i} = l_{t_j}$*, otherwise* $\mathscr{C}(t_i, t_j) = r$ *where* $t_i, t_j \in T$*, and* $r \in \Re$ *is a real value related with the size of data to be transmitted.*

Location and Cost Function. We have added location set L_T and cost function $\mathscr{C}$ to the original POCL plan. The L-POCL plan itself does not fully represent the workflow, since it does not maintain the information about process and execution conditions. However, it is still adequate to capture necessary information for constraint optimization. Here, location can be in any form that uniquely represents a physical place, such as IP address, host name, etc. A non-location-specific task has $\emptyset$ as its location. In relation to L_T, we introduce two more related sets $L_{|T|}$ and T^*_L.

- $L_{|T|}$: describes the number of tasks at the same location, i.e., $e \in L_{|T|}$ is a tuple $\langle l_i, n\rangle$, where l_i is the location, $t_i \in T$, and n is the number of tasks at location l_i.
- T_L^*: is a set of tasks $t \in T$ whose locations cannot be changed.

In terms of COP, the *hard constraint* must be satisfied strictly; in this scenario, that is the location of each task $t \in T_L^*$ cannot be changed. However, the *soft constraint* is preferred to be optimized. In this case, the concerned global function is the communication cost $\mathscr{C}$ between each task, for the reason that the data transmission between locations incurs communication costs. But when the tasks are at the same location, communication cost is marked to zero.

Consider the workflow as shown in Fig. 1, the L-POCL plan concepts involved are exemplified as follows:

$$
\begin{aligned}
T &= \{t_1, t_2, t_3, t_4, t_5, t_6\} \\
\succ_T &= \{\langle t_1, t_2\rangle, \langle t_1, t_3\rangle, \langle t_1, t_6\rangle, \langle t_2, t_3\rangle, \langle t_2, t_6\rangle, \langle t_3, t_6\rangle, \\
&\quad \langle t_4, t_2\rangle, \langle t_4, t_3\rangle, \langle t_4, t_5\rangle, \langle t_4, t_6\rangle, \langle t_5, t_6\rangle\} \\
\succ_C &= \{\langle t_1, t_2, c_1\rangle, \langle t_2, t_3, c_2\rangle, \langle t_3, t_6, c_3\rangle, \langle t_4, t_2, c_4\rangle, \langle t_4, t_5, c_4\rangle, \langle t_5, t_6, c_5\rangle\} \\
T_L^* &= \{t_1, t_4\} \\
L_T &= \{(l_1, t_1), (l_2, t_4), (\emptyset, t_2), (\emptyset, t_3), (\emptyset, t_5), (\emptyset, t_6)\} \\
L_{|T|} &= \{(l_1, 1), (l_2, 1), (\emptyset, 4)\} \\
\mathscr{C}(i,j) &= \begin{cases} 5 & \text{if } i = t_1 \text{ and } j = t_2, \\ 2 & \text{if } i = t_2 \text{ and } j = t_3, \\ 1 & \text{if } i = t_3 \text{ and } j = t_4, \\ 4 & \text{if } i = t_4 \text{ and } j = t_2, \\ 4 & \text{if } i = t_4 \text{ and } j = t_5, \\ 1 & \text{if } i = t_5 \text{ and } j = t_6 \end{cases}
\end{aligned}
$$

Note that L_T and $\mathscr{C}$ are also illustrated in Fig 2a Initial L_T table.

Multi-Agent L-POCL. We have shown that a workflow can be modeled by a L-POCL plan. However, the MAS configuration also requires information about agents and their task assignments. The task assignment is to be mapped with the corresponding location. Therefore, a multi-agent version of L-POCL is defined as follows:

Definition 2. *A **Multi-Agent L-POCL plan (ML-POCL)** is a tuple* $P = \langle A, T, \succ_T, \succ_C, L_T, \mathscr{C}, X\rangle$ *where*

- $\langle T, \succ_T, \succ_C, L_T, \mathscr{C}\rangle$ *is the embedded POCL plan,*
- A *is a set of the agents,*
- *Execution assignment* X *is a set of the tuples with the form* $\langle t, a\rangle$*, representing that the agent* $a \in A$ *is assigned to execute task* $t \in T$*.*

L_T and X are necessary components to construct a MAS configuration. Consider Listing 1.1, **hosts** are all locations found in L_T; **agents** are specified in X. L_T and X are mutually linked by tasks. Note that the naming scheme of an agent has no restriction as long as the name is unique.

4 Optimal MAS Configuration

This section describes how to obtain an optimal MAS configuration in terms of constraint optimization. A constraint optimization algorithm, in this case, is to minimize the soft constraint $\mathscr{C}$. The optimization reflects the change of L_T to an optimal one L_T^*. Then L_T^* is used to guide the construction of the optimal MAS configuration.

4.1 Constraint Optimization

The optimization algorithm migrates workflow tasks based on location. It uses the global cost function $\mathscr{C}$ as a guide by reducing the costly routes by minimizing the number of locations in the workflow.

Algorithm 1. Optimize $\mathscr{C}$

```
input  : L_T
output: L_T*
L_T* = L_T;
while there exists non-location-specific task in L_T* do
    foreach non-location-specific task t ∈ L_T* do
        let T_x be a set of location-specific tasks adjacent to t;
        if T_x = ∅ then continue;
        let t_x ∈ T_x be a task that maximizes C(t, t_x) or C(t_x, t).;
        update (∅, t) ∈ L_T* to (l_x, t) where l_x is the location of t_x.;
        if t_x maximizes C(t, t_x) then
            C(t, t_x) ← 0;
        else
            C(t_x, t) ← 0;
return L_T*;
```

Given a workflow as shown in Fig. 1, each arc is marked with the estimated size of data (presume in mega-bytes) to be transmitted between relevant tasks. Tasks t_1 and t_4 are at l_1 and l_2 respectively, and the locations of them cannot be changed since $t_1, t_2 \in T_L^*$. The corresponding location set L_T and cost function $\mathscr{C}$ can be presented in the tabular form as shown in Fig. 2a. Each header cell contains the combinations of each task $t_j (j = 1, \cdots, 6)$ and its location

$\mathscr{C}$	l_1,t_1	$\emptyset,t_2$	$\emptyset,t_3$	l_2,t_4	$\emptyset,t_5$	$\emptyset,t_6$
l_1,t_1	-	5	-	-	-	-
$\emptyset,t_2$	-	-	2	-	-	-
$\emptyset,t_3$	-	-	-	-	-	1
l_2,t_4	-	4	-	-	4	-
$\emptyset,t_5$	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2a Initial L_T table

$\mathscr{C}$	l_1,t_1	l_1,t_2	$\emptyset,t_3$	l_2,t_4	$\emptyset,t_5$	$\emptyset,t_6$
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	2	-	-	-
$\emptyset,t_3$	-	-	-	-	-	1
l_2,t_4	-	4	-	-	4	-
$\emptyset,t_5$	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2b Migrate t_2 to t_1

$\mathscr{C}$	l_1,t_1	l_1,t_2	l_1,t_3	l_2,t_4	l_2,t_5	$\emptyset,t_6$
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	0	-	-	-
l_1,t_3	-	-	-	-	-	1
l_2,t_4	-	4	-	-	0	-
l_2,t_5	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2c Migrate t_5 to l_2 and t_3 to l_1

$\mathscr{C}$	l_1,t_1	l_1,t_2	l_1,t_3	l_2,t_4	l_2,t_5	l_2,t_6
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	0	-	-	-
l_1,t_3	-	-	-	-	-	1
l_2,t_4	-	4	-	-	0	-
l_2,t_5	-	-	-	-	-	0
l_2,t_6	-	-	-	-	-	-

Fig 2d Optimized L_T^*

Fig. 2. Cost Function $\mathscr{C}$ Optimization

$l_i (i = 1, 2)$, note that $\emptyset$ means unspecified location. Moreover, '$-$' denotes the situation of unavailable connection. As *optimize*() is applied to the workflow, for instance, we get $\mathscr{C}(t_1, t_2) = 5$, where t_1 is at l_1 and t_2 is non-location-specific. The algorithm gradually evolves to the point L_T^* that every task is assigned with a optimal location. The algorithm performs the migration in two fashions: cost reduction and load balancing.

Cost Reduction Optimization. The algorithm searches for every non-location-specific task that is adjacent to at least one location-specific task. Firstly, $(\emptyset, t_2)$ is found with two adjacent location-specific tasks t_1 and t_4. As $\mathscr{C}(t_1, t_2) > \mathscr{C}(t_4, t_2)$, t_2 is migrated to the same location of task t_1, i.e., l_1. Tasks t_1 and t_2 are now at the same location l_1, then $\mathscr{C}(t_1, t_2)$ is updated to zero. Subsequently, the algorithm searches again and finds another non-location-specific task t_3, and then migrates it to the location of task t_2, i.e., l_1. The same principle goes on for non-location-specific task t_5, it is migrated to the location of t_4, i.e., l_2.

Load Balancing Optimization. As the cost reduction optimization goes on, the last non-location-specific task t_6 has two choices to migrate, t_3's location l_1 and t_5's location l_2, and both the values of $\mathscr{C}(t_3, t_6)$ and $\mathscr{C}(t_5, t_6)$ equal to 1. Now, let us consider the number of tasks at each location, indicated as $L_{|T|} = \{(l_1, 3), (l_2, 2), (\emptyset, 1)\}$. A simple load balancing advises to migrate task t_6 to location l_2. However, it also depends on other factors such as computing power which we do not cover in this paper, we will consider it for our future work. The final L_T^* is shown in Fig. 2d.

4.2 Obtaining MAS Configuration

This section discusses the way to obtain the optimal MAS configuration. The basic configuration rule is to assign one task to one agent, thus $|A| = |T|$ regardless of locations. But agents are related to the structure of L_T^* which has been attained in the previous steps. The execution assignment X of ML-POCL plan is also needed to obtain the desired configuration. Instead of starting out with one agent for one task, one agent is assigned to one location, thus initially $|A| = |L|$ regardless of tasks. Tasks that can be executed without having to wait for other tasks should be allocated with an agent. At each location $l \in L_T^*$, determine the agent allocation according to the workflow constructs (illustrated in Fig. 3):

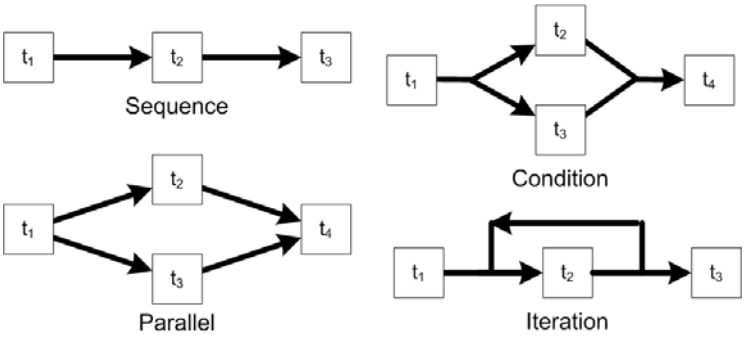

Fig. 3. Workflow Constructs

- **Sequence**: If task t_i has a successive task t_j defined in $\succ_T$, an agent a is assigned to t_i and t_j, s.t. $(t_i, a), (t_j, a)$.
- **Parallel**: If task t_i and t_j are not linked with temporal ordering as defined in $\succ_T$, two agents a_1 and a_2 are assigned to tasks t_i and t_j respectively, s.t. $(t_i, a_1), (t_j, a_2)$. For example in Fig. 1, if tasks t_2 and t_5 were at the same location, they are parallel.
- **Condition**: If task t_i is to choose either t_j or t_k exclusively, which means only one successive task will be executed, therefore an agent a is assigned to all t_i, t_j, t_k, s.t. $(t_i, a), (t_j, a), (t_k, a)$.
- **Iteration**: Iterative task is a sequential form of condition and loop. Let task t_i be the entrance to the iteration, T_i is the set of tasks to be repeated in a loop, t_j is the decision-making task, and t_k is the successive task of t_j, then an agent a is assigned to all tasks $t \in \{t_i, t_j, t_k\} \cup T_i$.

With respect to iteration, we cannot precisely calculate the number of iterations which will directly affect the total costs. But this is a factor that will influence every scheme of configuration proportionally.

5 Evaluation

In this section we demonstrate that the overall communication cost can be significantly reduced by taking advantage of the optimal MAS configuration strategy. For this purpose, we have simulated the execution of the randomly generated workflow in a distributed system composed of 5 hosts (locations). We randomly generate 20 workflows consisting of 10-20 tasks with random structures. In a workflow, 20-30% of all tasks are location specific. And each causal link is assigned with a random cost value between 1-5 (assumed mega-bytes). This simulation generates three MAS configurations for each workflow:

1. **Centralized configuration (C)**: All tasks run at one host.
2. **Equally distributed configuration (E)**: Equally assign a host with a number of tasks.
3. **Optimal configuration (O)**: The method we have proposed.

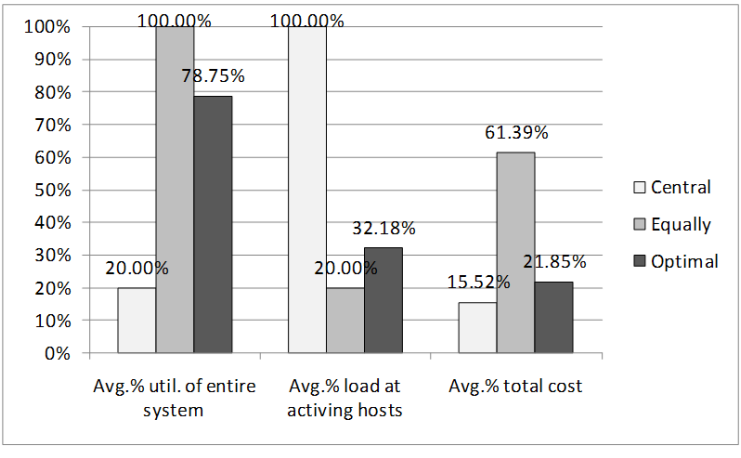

Fig. 4. Results of the Simulation

Based on the 20 workflows and 5 hosts, we calculate three evaluation values (the results are shown in Fig. 4):

1. **The average percentage of utilization of the entire system**: indicates the percentage of hosts being used to run the tasks. **C** uses only one host; **E** distributes all tasks to all hosts, so all hosts are utilized; and **O** uses only the hosts that have location-specific tasks, s.t., the number of active hosts is equal to $|L_T|$.
2. **The average percentage of load at the active hosts**: indicates the percentage of the number of tasks assigned to the active hosts. **C** runs all tasks at the only active host, s.t. 100%; **E** distributes all tasks to all hosts,

s.t. the number of tasks per host is $|T|/|\texttt{host}|$; and **O** uses only the hosts that have location-specific tasks, therefore, it is slightly higher than that of **E**, $|T|/|L_T|$.

3. **The average percentage of total cost**: indicates the amount of communication cost compared to the maximum cost in each workflow of the three configurations. In this simulation, we add one extra unit of cost for each causal link although all communications happened at the same host in order to demonstrate the activity that happens during the workflow enactment, otherwise **C** will always have zero cost. **C** communicates at a single host internally, then the cost is minimum; **E** communicates through out the system, so the cost is the highest among all; and **O** communicates only between selected hosts, therefore, the cost is between **E** and **C**.

The results of the simulation show that the proposed optimal MAS configuration both reduces the communication cost significantly and maintains a high level of system utilization. Its load balancing is better than that of centralized configuration, and obviously it cannot be better than that of equally distributed configuration as it does not use all of the hosts. With the proposed approach, we can achieve an optimal configuration that satisfies the hard constraints (location specific tasks) and optimizes the soft constraints (communication cost).

6 Conclusion and Future Work

We have presented a strategy to obtain an optimal MAS configuration for agent-enhanced data mining. The method utilizes the existing component in modern data mining tools, i.e., the workflow. The workflow is modeled with our proposed ML-POCL plan. The plan is then optimized and used to obtain an optimal MAS configuration. The result shows that the attained MAS configuration optimally reduces the communication cost and maintains a high level of system utilization.

However, there are a few issues yet to be improved. The constraint optimization process assumes all costs ($\mathscr{C}$) between tasks to be pre-defined. But in a more complex situation, the configuration should deal with cost function $\mathscr{C}$ dynamically since the environment may change and affect it. Another issue is that most WfMSs use data pipe-lining to pass data from one task to another, while global resource sharing scheme allows data production from one activity to be published to the naming directory service for later reference and re-use. This will help boost re-usability of the resource.

References

1. Bauer, T., Dadam, P.: Efficient Distributed Workflow Management Based on Variable Server Assignments. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 94–109. Springer, Heidelberg (2000)
2. Buhler, P.A., Vidal, J.M.: Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management 6(1), 61–87 (2005)

3. Cao, L., Gorodetsky, V., Mitkas, P.: Agent mining: The synergy of agents and data mining. IEEE Intelligent Systems 24(3), 64–72 (2009)
4. Cox, J., Durfee, E.: An efficient algorithm for multiagent plan coordination. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 828–835. ACM (2005)
5. Ehrler, L., Fleurke, M., Purvis, M., Savarimuthu, B.: Agent-based workflow management systems (WfMSs). Information Systems and E-Business Management 4(1), 5–23 (2006)
6. Huhns, M.N.: Agents as Web services. IEEE Internet Computing 6(4), 93–95 (2002)
7. Judge, D.W., Odgers, B.R., Shepherdson, J.W., Cui, Z.: Agent-enhanced Workflow. BT Technology Journal 16(3), 79–85 (1998)
8. Klusch, M., Lodi, S., Gianluca, M.: The role of agents in distributed data mining: issues and benefits. IEEE Comput. Soc. (2003)
9. Moemeng, C., Zhu, X., Cao, L.: Integrating Workflow into Agent-Based Distributed Data Mining Systems. In: Cao, L., Bazzan, A.L.C., Gorodetsky, V., Mitkas, P.A., Weiss, G., Yu, P.S. (eds.) ADMI 2010. LNCS, vol. 5980, pp. 4–15. Springer, Heidelberg (2010)
10. Moemeng, C., Zhu, X., Cao, L., Jiahang, C.: i-Analyst: An Agent-Based Distributed Data Mining Platform. IEEE (December 2010)
11. Odgers, B.R., Shepherdson, J.W., Thompson, S.G.: Distributed Workflow Coordination by Proactive Software Agents. In: Intelligent Workflow and Process Management. The New Frontier for AI in Business IJCAI 1999 Workshop (1999)
12. Savarimuthu, B.T., Purvis, M., Purvis, M., Cranefield, S.: Agent-based integration of Web Services with Workflow Management Systems. In: AAMAS 2005: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1345–1346. ACM, New York (2005)
13. Weld, D.S.: An Introduction to Least Commitment Planning. AI Magazine 15(4), 27–61 (1994)
14. Yoo, J.-J., Suh, Y.-H., Lee, D.-I., Jung, S.-W., Jang, C.-S., Kim, J.-B.: Casting Mobile Agents to Workflow Systems: On Performance and Scalability Issues. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, pp. 254–263. Springer, Heidelberg (2001)