Intra-Organizational Logistics Management Through Multi-Agent Systems

Eugene Santos Jr. (eugene@engr.uconn.edu) and Feng Zhang (fzhang@engr.uconn.edu)
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269

Peter B. Luh (luh@engr.uconn.edu)
Electrical and Computer Engineering
University of Connecticut
Storrs, CT 06269

Abstract. Compared to inter-organizational logistics management, the goal of intra-organizational logistics management is to maximize the profits of the whole company sometimes at the cost of its individual units' profits. While many agent-based models have been proposed for logistics management, most of these models use an auction approach. Thus, they are not suitable for intra-organizational logistics management. In this paper, we first formulate logistics management as a distributed resource allocation problem. Then we present our ongoing work on developing a multi-agent model for intra-organizational logistics management and using Lagrangian relaxation to decompose the problem into a set of subproblems. Our initial experimental results are very promising. We provide a detailed analysis of our results.

Keywords: logistics management, distributed resource allocation, multi-agent systems, Lagrangian relaxation

1. Introduction

According to a report by Forrester Research[1], business-to-business E-commerce will be a $1.3 trillion marketplace by 2003. The business-to-consumer marketplace will be roughly $120 billion. In fact, E-commerce has grown so quickly in the recent years that many companies have tried to invest in it. Some succeeded while others have failed. To summarize their experiences, the key issue is whether they can reduce cost under the increasing pressure of market globalization and intensive competition or not. Efficient logistics management is an effective tool to reduce cost, therefore very important to the success of a company focusing on E-commerce and E-business.

Though many traditional logistics management systems exist, they are not geared for E-commerce. The main reason is that they are off-line in nature while E-commerce is heavily on-line. As such, markets change very fast and, as a result, companies need to quickly react to such changes. Furthermore, customer demands show great variability. For example, Dell Computer manages to achieve "zero-inventory." Their approach is to capture market changes in a timely fashion and adapt to changes accordingly and very accurately[2].

In general, logistics management can be modeled as a distributed resource allocation problem. Such a problem consists of a set of suppliers and consumers. Suppliers provide resources, while consumers utilize resources to achieve specific goals, such as completing tasks or producing final products. After more than a decade of development in E-commerce, many logistics management models have been proposed and implemented. Some models are centralized, while others use a client/server approach (Kataoka et al., 1997; Caldwell, 1995). In recent years, researchers have proposed multi-agent based models (Walsh and Wellman, 1998; Walsh et al., 2000; Liu and Sycara, 1997; Sadeh et al., 1999; Henoch and Ulrich, 2000; Chen et al., 1999; Swaminathan et al., 1998). However, most of these models regard logistics management as an auction in which each entity tries to maximize its own profits. Such an approach is appropriate for inter-organizational logistics, which emphasizes maximizing individual profits since suppliers and consumers belong to different companies. However, many large E-commerce companies have their own supply chain. In this case, suppliers and consumers have a common goal of maximizing the profits of the entire company. The point here is that the maximized profits do not necessarily lead to the maximal benefits for each entity in the

[1] http://www.chiefexecutive.net/round/symbrand.htm
[2] http://www.lq.ca/issues/dec99/articles/article8.html

company. Therefore, an optimal schedule may be built by sacrificing certain entity profits.

To address intra-organizational logistics, some researchers have developed coordinating multi-agent logistics management models (Sadeh et al., 1999; Shen and Norrie, 1998; Kerschberg and Banerjee, 1999; Shen et al., 1999). (Sadeh et al., 1999) proposed MASCOT, a flexible multilevel architecture for coordinated supply chain planning and scheduling. (Shen and Norrie, 1998) proposed MetaMorph II, an architecture for enterprise integration and supply chain management, which is mediator-centric and agent-based. (Barbuceanu and Fox, 1996) proposed a KQML-based multi-agent coordination language for distributed and dynamic supply chain management. However, all these approaches are ad hoc, lacking any precise optimization model.

To overcome such limitations, we believe that it is necessary to model the resource allocation problem formally. Our approach utilizes Lagrangian relaxation (Luh et al., 1999; Zhao et al., 1999) to remove couplings between constraints so that the original problem can be separated into subproblems, corresponding to consumers, suppliers, tasks, and so forth. Separability allows us to allocate a different agent to solve each subproblem. If the solutions for these subproblems are compatible with each other, which means that constraints are not violated, we are done. Otherwise, these agents can exchange information with each other and search for an optimal way to satisfy the constraints. The exchanged information includes partial solutions, step values, subgradients, etc. With the decomposition of the original problem into subproblems, the computational complexity is also significantly reduced so that it is possible to get a near-optimal or optimal solution in a reasonable amount of time. Thus, we can put these agents on the Internet and let them respond to market changes in a timely fashion.

We begin with a discussion of related approaches in Section 2. Next, to provide a better understanding of intra-organizational logistics, we briefly present key issues in intra-organizational logistics in Section 3 and background on Lagrangian relaxation in Section 4. Based on our analysis, we formulate the problem in increasing order of complexity and present some experimental results (Sections 5 and 6). Finally, we discuss future work and provide concluding remarks.

2. Related Work

In this section, we describe related work on using multi-agent systems for logistics management. Generally, a multi-agent system consists of a set of software agents that operate continuously and autonomously in

a distributed environment. They try to achieve some goals in a smart way, which means that they can plan and decide the best action in a dynamic environment. Software agents can either work on their own, or cooperate with each other via a communication protocol, which can be developed in two different ways: using shared-memory and message passing.

Most existing approaches focus on the mapping between agents and activities. We call such a mapping, structure decomposition. For example, MASCOT (Sadeh et al., 1999) has lower-level agents as well as higher-level agents. Lower-level agents take the responsibility of planning and scheduling for a single facility over a short-term horizon. Higher-level agents aim at planning and scheduling across multiple facilities over a longer horizon. Other examples include (Zeng and Sycara, 1999; Yung and Yang, 1999; Liu and Sycara, 1997).

In addition to structure decomposition, some approaches consider scheduling decomposition. The work of (Yung and Yang, 1999) is closely related to our approach. They integrated multi-agent, Internet, and constraint satisfaction techniques to minimize operation costs and reduce the bullwhip effect in the supply chain. Their basic idea is to decompose the constraint satisfaction problem into subproblems and let propagation agents pursue optimized solutions for these subproblems via constraint propagation. Nevertheless, they only considered optimization in a static supply chain. (Liu and Sycara, 1997) considered the job shop scheduling problem in a similar fashion. Their assumption is that all jobs are known beforehand.

How do the agents interact and coordinate with each other? Most of these approaches use message passing techniques to facilitate agent interaction and coordination. An exception is shared memory used by agents to share information (Liu and Sycara, 1997). Some of those interaction protocols are based on KQML with some extensions (Barbuceanu and Fox, 1996; Yung and Yang, 1999). (Barbuceanu and Fox, 1996) used COOL, a KQML-based language, to represent and capture coordination knowledge. However, they did not address representations for constraints and partial solutions, which are critical for problem-solving agents to cooperatively solve an optimization problem.

Among these approaches, some are auction-based. For example, (Walsh and Wellman, 1998) proposed a market-based multi-agent model for decentralized task allocation. Agents will pay money for consumed goods and get paid for providing goods. All the bids for buying and selling goods are controlled by auction mediator agents. The negotiation process will start from the consumers buying goods and end at a valid solution. This model can be used to realize inter-organizational logistics

management. However, the negotiation process may not produce an optimal solution since each agent is self-interested.

Finally, similar to our goals, (Modi et al., 2001) also noticed the importance of properly formulating the resource allocation problem, but they take a distributed constraint satisfaction approach to solve the distributed resource allocation problem. This makes sense when only a few feasible solutions exist for a given problem or the solution quality does not matter. However, as we shall see, if the problem domain has preference over different solutions, our model is more suitable.

3. Issues

Before we can model intra-organizational logistics, we need to know its specific issues. Consider the following scenario: A manufacturing company consists of several factories and suppliers located at multiple locations. Each factory will utilize and consume raw materials, intermediate materials, and other resources provided by suppliers to make final products. Here, each factory can be viewed as a consumer. Since suppliers can only provide limited resources for any given period, the production needs of the factories may not be fully satisfied. The goal of intra-organizational logistics is to reasonably allocate resources so that the profits of the whole company are maximized. Depending on the following factors, the problem may be relatively easy or very complex:
Number of resource types: Typical resource types in the manufacturing domain are lathes, parts, and so on.
Number of suppliers for each factory or consumer: If each factory has only one supplier, the problem becomes easy. This is generally the case when a supplier is in charge of one geographical area. But with modern transportation means and networking, a supplier is no longer limited by its location. Therefore, the typical relationship between factories and suppliers is a many-to-many relation.
Resource properties: Different resource types are not fully separated. Two different types of resources may have overlapping functionalities. For example, two kinds of machines can produce the same parts, but with different costs. We call this property *resource exchangeability*. Based on resource exchangeability, we can allocate alternative resources if critical resources are in demand.
Task properties: A task can be the production activities in a given period. Generally, a deadline is set for each task. While missing deadlines should be avoided, being ahead of schedule can also be bad. The execution of a task needs a certain amount of resources. Sometimes, a task cannot be started unless all resources have been received. In other

cases, lack of resources only delays the execution time or degrades the quality of a task. Similarly, extra resources may accelerate the execution of a task or achieve a better goal. A task can either be a single step operation or consist of a sequence of stages. It is possible that some stages are critical. If a critical stage violates the deadline or cannot be continued for some reason, the whole task may be regarded as having failed. A consumer may execute more than one task. Several tasks may be related, such as having a common goal. Related tasks impose more constraints. For example, two tasks are required to begin at the same time.

Uncertainty: Tasks can arrive dynamically because of great variability in customer demands or the fact that resources may not be provided in time.

All these issues are inherent in intra-organizational logistics management and even more important when we look at the online nature of E-commerce. To handle all of these issues in a single model is clearly a challenging task. Traditional models only consider subsets of these factors and deal with the problem in a centralized manner, which means all the information is collected at one place for analysis. Only recently have agent-based techniques been used to solve the problem in a distributed environment.

4. Background on Lagrangian relaxation

Logistics management which can be modeled as a resource allocation problem deals with trying to satisfy multiple constraints. For example, there may be insufficient resources for executing all tasks simultaneously. Hence, optimally allocating resources to tasks can be viewed as a constrained optimization problem.

Lagrangian relaxation is a useful method for solving constrained optimization problems (Luh and Hoitomt, 1993). We all know that constrained optimization problems are harder than unconstrained ones. So the idea of the Lagrangian relaxation technique is to transform the originally constrained problem into an unconstrained one, for which a solution can be easily obtained, though infeasible for the original problem. By iteratively adjusting the unconstrained problem, a near optimal solution can be found in a reasonable amount of time. Here, the original problem is called the *primal* problem while the *Lagrangian dual* problem is the maximization of the relaxed problem.

Why are constrained problems so hard? The main reason is that constraints cross through multiple decision variables becoming a combinatorial optimization problem. With the relaxation of constraints,

decision variables get decoupled if the original problem is additive (separable). As a result, the relaxed problem can be solved in polynomial time. By using Lagrange multipliers to decouple and relax complicated constraints, the primal separable problem is relaxed into individual subproblems which are minimized respectively given a set of multipliers. The Lagrangian dual function, whose value is based on the minimization of subproblems, is then maximized over iterations by adjusting the multipliers. It is shown that the degree with which constraints are violated while solving the relaxed subproblems will be reduced through careful adjustment of the multipliers. Also, the dual function provides a *lower bound* to the optimal primal cost. The difference between a feasible solution cost and the maximum dual cost is called the *duality gap*. Since the dual solution is generally infeasible due to violation of some constraints, heuristics can be used to construct a feasible solution based on the infeasible dual solution.

Given that the dual function is non-differentiable in some cases, like integer optimization, subgradient methods can be used to maximize the dual function. The subgradient direction is obtained by minimizing all the subproblems and then updating the multipliers along the subgradient direction. We now show the Lagrangian relaxation framework based on the subgradient method.

A decomposable integer optimization problem can be described as follows:

$$\min_x \sum_i J_i(x_i) \tag{1}$$

subject to

$$Ax \leq b, x_i \in X_i \subset Z^{n_i}, i = 1, ..., I,$$

where $J_i(x)$ are convex (possibly nonlinear) functions, $x = [x_1, x_2, ..., x_I]^T$ is the $n \times 1$ decision variable with $n = \sum_i n_i$, and X_i is the domain of x_i. The $m \times n$ matrix A is of the form $[a_1, a_2, ..., a_I]$, in which a_i is an $m \times n_i$ vector, and b is an $m \times 1$ vector.

Note that the m constraints $Ax \leq b$ couple the decision variables. Through relaxation, these m constraints are decoupled so that the primal problem can be decomposed into subproblems. The Lagrangian relaxation of the problem is given by

$$\min_x L, \text{ with } L = [\sum_i L_i(x_i) + \lambda^T(Ax - b)], \tag{2}$$

where λ is an $m \times 1$ vector of Lagrange multipliers. This can be written in terms of individual subproblems

$$\min_{x_i} L_i, \text{ with } L_i = [J_i(x_i) + \lambda^T(a_i x_i)] \tag{3}$$

Let L_i^* be the minimal subproblem cost. The Lagrangian dual problem is:

$$\max_{\lambda} L, \text{ with } L = \sum_i L_i^* - \lambda^T b \tag{4}$$

When the subgradient method is used to maximize the dual function, multipliers λ are adjusted according to the following formula:

$$\lambda^{k+1} = \lambda^k + s^k g(\lambda^k), \tag{5}$$

where s^k is the step size, and $g(\lambda^k)$ is the subgradient of L at λ^k, given by:

$$g(\lambda^k) = Ax^k - b = \sum_i a_i x_i^k - b, \tag{6}$$

where

$$x_i^k = arg \min_{x_i} [J_i(x_i) + (\lambda^k)^T (a_i x_i)]. \tag{7}$$

It has been proven that the subgradient method will update multipliers along the direction towards λ^*, which corresponds to a maximum dual cost (Zhao et al., 1999).

Note that it is easy to use the Lagrangian relaxation technique centrally since we can implement a program to sequentially solve these decomposed subproblems in each iteration and derive a feasible solution thereafter via heuristics, such as the "list scheduling" heuristic used in this paper (Luh and Hoitomt, 1993). However, we are more interested in employing multi-agents to realize the Lagrangin relaxation technique. Basically, the dual problem can be solved efficiently, with each subproblem taken care of by a different agent. Another agent can use heuristics to derive feasible solutions from infeasible solutions.

5. Formulation

Our approach is different from existing approaches presented earlier in that we take into account a precise optimization model. In this section, we classify the resource allocation problem into four models, each of which considers different subsets of issues discussed in Section 3. E-commerce activities, in particular, our intra-organizational supply chain management, have as their central component interactions between entities taking the role of consumers and others taking the role of suppliers accordingly. Some activities are relatively simple involving only one resource type to be shared. Other activities are more complex, often having uncertainties due to the dynamic nature of the planning and scheduling problem. The following four models reflect these activities in increasing order of complexity. The first three can

be applied in a static environment while the last one is targeted for a dynamic environment. Each model can be represented by an integer programming formulation. By using Lagrangian relaxation, we intend to decompose a resource allocation problem into subproblems, each of which can be solved by a specific agent. If the solutions for these subproblems are compatible with each other, which means that constraints are not violated, we are done. Otherwise, these agents can exchange information with each other until a near-optimal or optimal solution is found. The exchanged information includes partial solutions, step values, subgradients, etc. Lagrangian relaxation approach is iterative. After exchanging information, these agents will have newly adjusted Lagrange multiplier values so that they can solve task-level subproblems again. Then they will exchange information to see whether they need further coordination. Sometimes, it needs a long time (maybe forever) to make the dual solution feasible. Therefore, we stop the scheduling process after a fixed number of iterations and derive a feasible solution based on the best found dual solution.

5.1. Single Resource Type/Single Supplier/Single Task

In this model, we assume that only one kind of resource exists in the system, and each consumer requests resources from a corresponding supplier to do a single task. We also fix the cost of using a unit of resource. Therefore, there is no need to consider the resource cost when we do task scheduling and the goal is to maximize on-time task execution. Here, on-time task execution can be understood from two aspects. On one hand, it is desirable to complete a task before its due date. On the other hand, we should not start or complete a task too early.

5.1.1. *Problem formulation*

Since each consumer has only one task, we consider only tasks and suppliers here. Let the number of tasks be N_c and the number of suppliers be N_s. For each task $i = 1, ..., N_c$, b_i represents the starting time, c_i represents the completion time, p_i represents the execution time, d_i represents the deadline for task i, s_i represents the supplier from which task i will get resources ($s_i \in [1, N_s]$), and r_i represents the needed number of resources. Also, we use T_i to measure the delay of task i ($T_i = \max(0, c_i - d_i)$). Because some tasks may be more important than others, it is not desirable to delay these tasks. To reflect this factor, we use w_i to represent the importance of each task. The higher the value of w_i is, the more important the task is. For a given task due date d_i, a desired task start time is: $d_i - p_i + 1$. We use E_i to measure

the earliness of task i ($E_i = \max(0, d_i - p_i + 1 - b_i)$) and weight β_i to account for the cost of being early. We can define the objective function as:

$$\min \sum_i (w_i T_i^2 + \beta_i E_i^2) \tag{8}$$

At any given time t, the total granted resources from one supplier must be less than its capacity. We use $N_j (j \in [1, N_s])$ to represent the amount of resources that supplier j has. We assume the time horizon T is large enough to complete all the tasks. The resource capacity constraints can be expressed as:

$$\sum_i \delta_{ijt} \cdot r_i \leq N_j, \tag{9}$$

where $j \in [1, N_s], t \in [1, T]$, and

$$\delta_{ijt} = \begin{cases} 1 & \text{If } s_i = j \wedge b_i \leq t \leq c_i \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

In addition, the following processing time constraints must be satisfied:

$$c_i - b_i + 1 = p_i, \text{ for } i = 1, ..., N_c \tag{11}$$

In the above formulation, the task starting times b_i, the task completion times c_i, task tardiness T_i and earliness E_i, and the active tasks at one time are unknown. But when the task starting times b_i are known, the other variables can be derived. Therefore, the scheduling problem is to select the task starting times b_i^* to optimize on-time task execution, subject to the resource capacity constraints and the processing time constraints.

For example, consider the following E-business logistics problem: assume 2 suppliers and 3 task groups. Each supplier has one machine. Each task group is required to produce some parts.
-$N_s = 2, N_1 = 1, N_2 = 1, N_c = 3$
-Task1: Produce part 1. $r_1 = 1, d_1 = 5, p_1 = 3, s_1 = 1, w_1 = 7$
-Task2: Produce part 2. $r_2 = 1, d_2 = 5, p_2 = 5, s_2 = 2, w_2 = 10$
-Task3: Produce part 3. $r_3 = 1, d_3 = 6, p_3 = 4, s_3 = 1, w_3 = 6$

Because each task needs one machine but only two machines are available, at most two tasks are allowed to be executed at the same time. Both task 1 and task 2 will request resources from supplier 1 so they cannot be executed simultaneously. Now consider two possible schedules.

a: First execute task 2 and task 3, and then execute task 1. We have: $b_2 = 1, c_2 = 5, b_3 = 1, c_3 = 4, b_1 = 5, c_1 = 7, T_2 = T_3 = 0, T_1 = 2$. The objective function will have the value 28.

b: First execute task 1 and task 2, and then execute task 3. We have: $b_1 = 1, c_1 = 3, b_2 = 1, c_2 = 5, b_3 = 4, c_3 = 7, T_1 = T_2 = 0, T_3 = 1$. The objective function will have the value 6.

Apparently, the second schedule is a better one. In fact, it is the optimal.

5.1.2. *Solution methodology*

Note that the processing time constraints are associated with individual tasks, and the objective function is also task-wise additive. Only the resource capacity constraints couple across tasks. Lagrangian relaxation is employed to relax the resource capacity constraints so that the original problem can be decomposed into a set of subproblems.

Since there is a capacity constraint for each supplier at time t, the Lagrange multiplier π_{jt} is used to relax the capacity constraint for supplier j at time t. As a result, we get the relaxed problem:

$$\min_{b_i}\{\Sigma_i(w_i T_i^2 + \beta_i E_i^2) + \Sigma_{jt}\pi_{jt}(\Sigma_i \delta_{ijt} \cdot r_i - N_j)\} \tag{12}$$

subject to the processing time constraints. The dual problem is:

$$\max_{\pi \geq 0} L, \text{ with } L = \{-\Sigma_{jt}\pi_{jt}N_j + \min_{b_i} \Sigma_i(w_i T_i^2 + \beta_i E_i^2 + \Sigma_{jt}\pi_{jt}\delta_{ijt} \cdot r_i)\}, \tag{13}$$

in which π_{jt} is non-negative real number, and $\pi = [\pi_{jt}]$. The dual problem is also subject to the processing time constraints. Then, the task-level subproblems are:

$$\begin{aligned} \min_{b_i} L_i, \text{ with } L_i &= \{w_i T_i^2 + \beta_i E_i^2 + \Sigma_{jt}\pi_{jt}\delta_{ijt} \cdot r_i\} \\ &= \{w_i T_i^2 + \beta_i E_i^2 + \Sigma_{t=b_i}^{b_i+p_i-1}\pi_{s_i t} r_i\}, \end{aligned} \tag{14}$$

where the processing time constraints have been incorporated. Basically, an iterative approach is used to solve the problem. At each iteration, given current multipliers, either all subproblems (e.g. in the subgradient method) or only parts of them (e.g. in the surrogate subgradient method (Zhao et al., 1999)) are minimized. Thus, the dual problem cost can be attained. Next, multipliers will be adjusted according to (5) and a new iteration will begin. Since the duality gap will get reduced over the iterations, some stop criteria can be used such as the iteration number, duality gap, etc. The computational complexity for solving a subproblem is linear in the time horizon T. So the total computational complexity for solving all subproblems at each iteration is $\mathrm{O}(N_c \times T)$.

Let L_i^* be the minimal cost of subproblem i, which can be calculated via enumerating all possible starting times, given current multipliers.

Then, in the case of using the subgradient method, the overall dual problem becomes:

$$\max_{\pi \geq 0} L, \text{ with } L = \{-\Sigma_{jt}\pi_{jt}N_j + \Sigma_i L_i^*\}, \tag{15}$$

The reason why we use various subgradient methods is that the dual problem is non-differentiable here. The following formula is used to adjust multipliers:

$$\pi^{n+1} = \pi^n + s^n g(\pi^n), \tag{16}$$

in which n and $n+1$ are iteration indexes, $s^n(\geq 0)$ is the step size, and $g(\pi^n)$ is the subgradient of L with respect to π. At time t, the j^{th} component of $g(\pi^n)$ is $\sum_i \delta_{ijt} r_i - N_j$.

Although the dual solutions satisfy the processing time constraints, they violate the capacity constraints. Fortunately, by adjusting multipliers each iteration, violations only occur at a few time slots. Therefore, we use the "list scheduling" heuristic to get feasible solutions based on the dual solutions. First, we sort all the tasks in an increasing order of starting times. If there are not enough resources for the tasks scheduled at time t, tasks leading to high penalties will be assigned the needed resources while other tasks are delayed by one time slot and compete for resources at time $t+1$. This process of comparison, assignment and delay will last until all tasks are scheduled with capacity constraints satisfied. For each delay, the task needs to be inserted at a new location in the list, which in worst case can be done in $O(N_c)$. Consider that each task will be assigned a suitable starting time after several trials (worst case). The complexity of the "list scheduling" heuristic is $O(N_c^2 + N_c \log N_c)$, in which $N_c \log N_c$ is the sorting time.

5.2. Multiple Resource Types/Single Supplier/Single task

In general, there are many kinds of resources in more complex E-commerce domains and tasks need several kinds of resources. Consider the previous example. Now we extend it by including another machine type and assuming that each task needs such a machine as well. Such an example can be solved by the following model.

To avoid being overly complicated, we consider a single supplier and a single task per consumer in this model. Therefore, the objective is still to optimize the overall on-time task execution.

Let N_{rt} be the number of resource types in the system. Let $r_{il}(i \in [1, N_c], l \in [1, N_{rt}])$ be the number of resource l that task i needs. We denote the number of resource l that supplier j has by N_{jl}. The new model can be formulated as follows:

The objective function and the processing time constraints are the same as (8) and (11) in the above case.

The resource capacity constraints (needed to be satisfied at time t):

$$\sum_{i=1}^{N_c} \delta_{ijlt} \cdot r_{il} \leq N_{jl}, j \in [1, N_s], l \in [1, N_{rt}], t \in [1, T] \tag{17}$$

$$\sum_{j=1}^{N_s} \delta_{ijlt} = \begin{cases} 1 & \text{If } b_i \leq t \leq c_i \\ 0 & \text{Otherwise} \end{cases} \tag{18}$$

where $\delta_{ijlt} = \begin{cases} 1 & \text{If task } i \text{ gets resource } l \text{ from supplier } j \text{ at time } t \\ 0 & \text{Otherwise} \end{cases}$

The scheduling problem is to select the task starting times b_i^* to minimize the overall tardiness and earliness penalties, subject to the resource capacity constraints and the processing time constraints.

While this model contains multiple resource types, its additive nature is not changed. And it can be handled in the same way by using Lagrangian relaxation as in Model 1. The computational complexity for solving the dual problem is $O(N_c \times N_{rt} \times T)$. Since multiple resource types are considered in this model, a task may need any type of resource. To derive a feasible solution via the "list scheduling" heuristic, we need to know for each task, whether its resource request at time t are satisfiable or not. This can be known in $O(N_{rt})$. So the complexity of the heuristic is $O(N_c(N_c + N_{rt}) + N_c \log N_c)$.

5.3. Multiple Resource Types/Multiple Suppliers/Single task

The drawback of the previous models is that a consumer cannot execute a task until the corresponding supplier has enough resources. Still consider the previous example, tasks 1 and 2 cannot be executed at the same time because their corresponding supplier 1 has only one machine. If the other supplier has one extra machine and either task 1 or task 2 can use this machine, we can get a better solution. Therefore, by allowing a consumer to request resources from multiple suppliers, this model is more flexible and can result in better task schedules.

5.3.1. *Problem formulation*

We assume different costs for using resources from different suppliers. Therefore, the overall aim is to minimize tardiness and earliness penalties and resource cost as a whole. We formulate the model as follows:

The objective function:

$$\min \sum_i (w_i T_i^2 + \beta_i E_i^2 + \sum_{jlt} e_{it} r_{ijl} c_{rjl}), \tag{19}$$

where e_{it} is one if task i is active (in execution) at time t and zero otherwise; r_{ijl} represents the number of resource l that task i gets from supplier j; $c_{rjl}(j \in [1, N_s], l \in [1, N_{rt}])$ is the cost of using unit resource l from supplier j per unit time.

The resource capacity constraints are:

$$\sum_{i=1}^{N_c} e_{it} r_{ijl} \leq N_{jl}, j = 1, ..., N_s, l = 1, ..., N_{rt}, t = 1, ..., T, \sum_j r_{ijl} = r_{il}, \tag{20}$$

where r_{il} is the number of resource l needed by task i.

The processing time constraints are the same as (11).

The scheduling problem is to select the task starting times $\{b_i^*\}$ and the obtained resources l from supplier j $\{r_{ijl}^*\}$ to minimize the overall tardiness and earliness penalties and resource cost, subject to the resource capacity constraints and the processing time constraints. Similarly, the standard Lagrangian relaxation technique can be utilized to get a near-optimal schedule in this model. The computational complexity for solving the dual problem is $O(N_c \times N_{rt} \times N_s \times T)$ since minimization of each subproblem will take $O(N_{rt} \times N_s \times T)$ in worst case. Generally, each consumer will have a couple of preferred suppliers and each task may not need all the different types of resources. So the average case complexity may be fairly small. To derive a feasible solution via the "list scheduling" heuristic, we need to know for each task, whether its resource request at time t is satisfiable or not. This can be determined in $O(N_{rt} \times N_s)$. Thus, the complexity of the heuristic is $O(N_c(N_c + N_{rt} \times N_s) + N_c \log N_c)$.

5.4. MR/MS/MJ/MNS/AR/DR/LR/DJ/ER

The earlier models all rely on assumptions that hold only in static environments. Ultimately, for real-world E-commerce problems, resources can arrive dynamically while other resources may be damaged or lost over time. If two resources have similar properties, one may be used if the other is not available for a task. Also, a task may have many steps. Since each step has its own resource requirements, it is not necessary to receive all resources before the execution of the first step. Instead, we can give resources to a step just before its execution. Tasks can also arrive dynamically. When a new task comes in, there may already be tasks in execution. We can either stop all tasks in execution and

reschedule, or just consider the remaining tasks plus the new one while allowing currently active tasks complete. We now address these issues as follows:
-MR: Multiple resource types
-MS: Multiple suppliers for each consumer
-MJ: Multiple tasks for each consumer
-MNS: Each task may have many steps
-AR: Allow alternative resources
-DR: Resources arrive and depart dynamically
-LR: A task can be started with extra cost if there are not enough resources.
-DJ: Tasks arrive dynamically.
-ER: Allocating extra resources to a task can shorten its execution time.

5.4.1. *Problem formulation*
Let N_{jlt} be the number of resource l available from supplier j at time t. To establish a relation between two alternative resources, let $\rho_{l'l}$ be the equivalent number of resource l for a unit of resource l'. If two resources l and l' have no relation, we let $\rho_{l'l} = 0$. Let N_{is} be the number of steps of task i. Two adjacent steps of a task may have time limits. For example, the next step must begin in time γ after the completion of the current step. We use $\gamma_{nis}/\gamma_{xis}$ to represent the minimal/maximal interval between step s of task i and its next step. When a consumer has multiple tasks, it may execute them sequentially or in parallel. Here, we assume that a consumer can only run one step at any time and run tasks sequentially. In the future, we will address the issue of executing several steps concurrently.

Given that a consumer can have multiple tasks, let o_i and N_c be the corresponding consumer for task i and the number of consumers respectively ($o_i \in [1, N_c]$). In general, a consumer will try to get all needed resources before executing a task. However, it often happens that only some of the resources are available. In this model, the consumer can either 1) try again later but the task deadline may be violated; or 2) start the task if only "unimportant" resources are unavailable. Of course, doing so may increase costs or lead to longer execution times. If a task consists of many steps, resources needed in different steps can also be different. So, there is a tradeoff between waiting for all resources and requesting resources on the fly. In this model, we use β_{isl} to represent the cost of executing step s of task i if one unit of resource l is lacking.

Since tasks can arrive dynamically, let N_{ct} be the number of tasks at time t. The arrival time of task i is denoted by a_i. We have: $a_i > t \Rightarrow$

$N_{ct} < i$. We assume tasks are preemptible. When a new task needs to be executed right away and there are not enough resources, currently active tasks may be stopped if the new task is very important. To stop an active task increases its cost. Let θ_{is} be the penalty for interrupting step s of task i. One task may be interrupted several times during its execution. We assume each step must be an atomic operation. Therefore, if one step is interrupted, it must restart from scratch. Let INT_{is} be the repeated times of task i at step s. $INT_{is} - 1$ is the number of interruptions of task i at step s. For each task, $b_{is\alpha}, p_{is\alpha}, c_{is\alpha}$ represent the starting time, the processing time and the completion time of step s of task i after being interrupted $\alpha-1$ times ($s \in [1, N_{is}], \alpha \in [1, INT_{is}]$).

Since this model involves uncertainties, like uncertain arrival time, uncertain execution time, uncertain amount of resources, and so on, it is nearly impossible to always get an optimal schedule. Instead, a reasonable goal should be to minimize the expected cost. Using a stochastic programming approach, the model can be formulated as follows:

The objective function:

$$\min \; E[\textstyle\sum_i (w_i T_i^2 + \beta_i E_i^2 + \sum_s \theta_{is} INT_{is} + \sum_{sl} \beta_{isl}(r_{isl} - \sum_{jl'} r_{ijsll'} \rho_{l'l}) + \sum_{jsll't} e_{ist} r_{ijsll'} c_{rjl'})], \tag{21}$$

where $c_{rjl'} (j \in [1, N_s], l' \in [1, N_{rt}])$ is the cost of using unit resource l' from supplier j per unit time; e_{ist} is one if step s of task i is active (in execution) at time t and zero otherwise; $r_{ijsll'}$ is the number of resource l' that task i gets from supplier j for replacing resource l during step s; r_{isl} represents the number of resource l needed at step s of task i; $T_i \in [0, c_{i,N_{is},INT_{is}} - d_i]$.

Because tasks cannot be executed before their arrival, the arrival time constraints must be satisfied:

$$a_i \leq b_{i1\alpha}, \text{ for } i \in [1, N_{cT}], \alpha \in [1, INT_{is}]. \tag{22}$$

The exact resource capacity constraints are very difficult to handle for all possible schedules because of complexity and uncertainty. Therefore, we use expected values to model them:

$$E[\sum_{isl} e_{ist} r_{ijsll'}] \leq N_{jl't}, \text{ for} j \in [1, N_s], l' \in [1, N_{rt}], t \in [1, T]. \tag{23}$$

Insufficient or extra resources may influence the quality of a task execution. Let σ_{isl} be the influence of insufficient resources and extra resources on the execution time of task i at step s. Let p_{is} be the intended execution time of task i at step s. The actual execution time of task i at step s must satisfy:

$$p_{is\alpha} \; = \; \sum_l (\sigma_{isl}(r_{isl} - \sum_j \sum_{l'} r_{ijsll'} \rho_{l'l})) + p_{is}, \text{ for } i \in [1, N_{cT}], \tag{24}$$

$$s \in [1, N_{is}], \alpha \in [1, INT_{is}]$$

The processing time constraints are at the step level:

$$c_{is\alpha} - b_{is\alpha} + 1 < p_{is\alpha}, \text{ for } i \in [1, N_{cT}], s \in [1, N_{is}], \alpha \in [1, INT_{is} - 1] \quad (25)$$

and,

$$c_{is\alpha} - b_{is\alpha} + 1 = p_{is\alpha}, \text{ for } i \in [1, N_{cT}], s \in [1, N_{is}], \alpha = INT_{is} \quad (26)$$

The operation precedence constraints state that the succeeding operation of the current operation of a task must be started after the current operation has been completed plus a timeout value:

$$\gamma_{mis} \leq b_{is'1} - c_{is,INT_{is}} - 1 \leq \gamma_{xis}, \text{ for} i \in [1, N_{cT}], s \in [1, N_{is}], s' = s + 1 \quad (27)$$

Since two tasks of a consumer are executed sequentially, the switching time constraints need to be satisfied between the last step of the previous task and the first step of the current task:

$$\delta_{ik}(c_{i,N_{is},INT_{is}} + s_{ik} + 1 - b_{k11}) \leq 0, \quad (28)$$

where

$i, k \in [1, N_{cT}], i \neq k, o_i = o_k$;

$\delta_{ik} = \begin{cases} 1 & \text{If task } k \text{ is started after task } i \text{ has been started} \\ 0 & \text{Otherwise} \end{cases}$

The scheduling problem is to select suitable operation starting times $b^*_{is\alpha}$ and resources $r^*_{ijsll'}$ based on the realization of random events to minimize the overall cost, including tardiness/earliness penalties, resource unavailability penalties, and resource cost.

5.4.2. *Solution methodology*

Based on the assumption of the switching time being a constant, we can replace the switching time constraints by the following expected consumer capacity constraints:

$$E[\sum_{o_i=k} \sum_{s} e_{ist}] \leq 1, i = 1, ..., N_{cT}, k = 1, ..., N_c, t = 1, ..., T \quad (29)$$

By applying Lagrangian multipliers π_{jlt} and λ_{kt} to relax expected resource capacity constraints and consumer capacity constraints respectively, we obtain the following relaxed problem:

$$\min_{b_{is}, r_{ijsll'}} L, \text{ with } L = E[\sum_{i}(w_i T_i^2 + \beta_i E_i^2 + \sum_{s} \theta_{is} INT_{is} + \sum_{sl} \beta_{isl}(r_{isl} - \sum_{jl'} r_{ijsll'} \rho_{l'l}) +$$

$$\sum_{jsll't} e_{ist} r_{ijsll'} (c_{rjl'} + \pi_{jl't}) + \sum_t \lambda_{o_i,t} e_{ist})] - \sum_{kt} \lambda_{kt} - \sum_{jl't} \pi_{jl't} N_{jl't}, \quad (30)$$

subject to the arrival time constraints, operation precedence constraints, and processing time constraints.

This can be written in terms of individual subproblems after regrouping related terms:

$$\min_{b_i s, r_{ijsll'}} L_i, \text{ with } L_i = E[\sum_i (w_i T_i^2 + \beta_i E_i^2 + \sum_s \theta_{is} INT_{is} + \sum_{sl} \beta_{isl} (r_{isl} - \sum_{jl'} r_{ijsll'} \rho_{l'l}) + \sum_{jsll't} e_{ist} r_{ijsll'} (c_{rjl'} + \pi_{jl't}) + \sum_t \lambda_{o_i,t} e_{ist})] \quad (31)$$

subject to the above constraints.

Let L_i^* be the minimal subproblem cost for task i. The Lagrangian dual problem is:

$$\max_{\pi,\lambda} L_D, \text{ with } L_D = \max_{\pi,\lambda} [\sum_i L_i^* - \sum_{jl't} \pi_{jl't} N_{jl't} - \sum_{kt} \lambda_{kt}] \quad (32)$$

To solve the dual problem, we can use the backward stochastic dynamic programming approach given in (Luh et al., 1999).

6. Empirical results

Through experimentation, we will demonstrate that our approach often leads to near-optimal or optimal solutions in a reasonable amount of time. We have decided to simulate our multi-agent subproblems on a single processor in order to more readily analyze performance. Ultimately, each subproblem can be processed in parallel and in a distributed fashion. This will be critical for many domains, where fast response to varying situations is desired.

Given a problem, if the optimal solution is known, then evaluating solutions generated by an algorithm is straightforward. For small problems, it is possible to obtain the optimal solution via brute force enumerations. However, this is not realistic for large-scale problems. Our Lagrangian relaxation technique does provide a lower bound (the dual cost) for the optimal solution. If the duality gap is very small, we know that the solution we found is near-optimal. For example, if

the primal cost is 5440.00 and the dual cost 5402.34, the duality gap is 0.70%, which means the primal solution is within 0.70% of the optimal. For some test cases, we obtain very small duality gaps. Unfortunately, in many cases, the duality gaps are too big. Based on our analysis, we found that in cases with large duality gaps, the dual cost is actually a poor lower bound. We performed this analysis by studying problems amenable to brute force methods to determine their optimal solutions. For example, a problem in Model 1 or Model 2 with ten tasks can be solved in about ten minutes on a Pentium3-866MHz machine using brute force. A longer time is needed for problems in Model 3: about forty minutes. We then compared our solution with the optimal. We use *real gap* to denote the difference from the primal solution cost obtained by our Lagrangian relaxation to the optimal one. If the real gap is within 10%, we regard the primal solution as near-optimal. We conclude that our approach has a good chance of finding a near-optimal or optimal solution, although the duality gap may be large.

6.1. Test Bed

We implemented a scheduling package to solve problems belonging to the first three models presented in Section 5. The test cases we use are randomly generated. The four main parameters in generating test cases are due dates, tardiness weights, processing times and requested resource numbers. To simplify, we ignore earliness weights in these test cases. However, we consider another parameter, available resources for each supplier, because requested resources are restricted by it. Given a specific problem, its tasks are reflected by some distribution on due dates, tardiness weights, processing times, and requested resources. In this paper, we assume either a uniform distribution or a Gaussian distribution or a constant value for these parameters. We include a Gaussian distribution in order to model peak demand. Many combinations exist for selecting these parameters. We explore three combinations for our test bed:

1. Comb-1: Due dates, processing times, and tardiness weights are uniformly distributed while available resource number is a constant. Requested resource numbers are either constant or uniformly distributed.

2. Comb-2: Due dates, processing times, tardiness weights, available resource numbers, and requested resource numbers are all uniformly distributed.

3. Comb-3: Due dates satisfy the Gaussian distribution. Processing times, tardiness weights, available resource numbers and requested resource numbers are uniformly distributed.

We created three sets of test cases for each of the combinations. The parameter values are shown in Table I, where S-Num, SN-Per-Task, RT-Num, AR-Num and P-Time are respectively the number of suppliers, the number of suppliers providing resources for each task, the number of resource types, resources provided by each supplier, and the duration time of each task. In Comb-2 and Comb-3, available resources are uniformly distributed. So the number of requested resources for each task is determined by available resources. We use Res-On-Sup to represent the maximal number of resources that can be assigned to each task. For each parameter setting (shown in Table I), *fifty* test cases are created. We measure the average performance of our approach over these fifty test cases. The problem space is determined by the number of tasks, which is $N_c!$. Here, we fix the number of tasks for our tests to 10.

6.2. Experimental Analyses

All experiments were conducted on a group of DELL PCs (Pentium II 333MHz or above). In all test cases, we initialize all Lagrange multipliers to zero. Next, we use the subgradient method to solve the dual problem iteratively. A decision now had to be made: how many iterations should each experiment run? We would ideally use a multi-agent model to solve our resource allocation problem in parallel. We also would like to spend only a small amount of elapsed time (≤ 1 minute) to compute a good feasible solution. The separability of our formulation makes it possible to solve each problem through parallelization. From our observations, the average time for solving a test case with 10 tasks is about thirty seconds for 1,000 iterations on a Pentium 3-866MHz PC. By applying parallelization, we should finish this in a few seconds.

Recall that the "list scheduling" heuristic is used to derive a feasible schedule from the dual solution. Operations or tasks are first sorted in ascending order of starting times given by the dual solution and then scheduling according to this order. If several operations or tasks are scheduled at the same time slot but no enough resources are available, a heuristic is used to determine which operations or tasks should be delayed by one time unit. The original heuristic (Luh and Hoitomt, 1993) is based on comparing the incremental changes in weighted tardiness penalties if each of them is delayed by one time unit. Those with higher changes are scheduled first. Others will be delayed. Based on our preliminary results, we found that the original heuristic did

not work well on some test cases, shown in Table II, which contains the maximum, minimum, mean, and standard deviation of real gaps for each group of test cases. In this table, "# of successful cases" is the number of cases, whose solutions obtained by our model are near-optimal (within 10% of the optimal solution). We next replaced the original heuristic by the WSPT/CR heuristic, which is a combination of weighted shortest process time and critical ratio (Chen et al., 2001). The overall performance of the WSPT/CR heuristic is much better than the original heuristic (Table III). Ori-Heu and WSPT-Heu are the cost of the best found primal solution given by using the original heuristic and the cost of one obtained by the WSPT/CR heuristic. We also noticed that on several test cases, the new heuristic leads to worse primal solutions than those given by the original heuristic (Table IV). So we combine them together simply in the "list scheduling" heuristic, switching between these two heuristics every iteration, for our experiments. The combined heuristic does avoid very worse primal solutions compared to either the original heuristic or the WSPT/CR heuristic (Table V). However, the combined heuristic works worse on several cases compared to using the WSPT/CR heuristic or the original heuristic alone (Table VI). This is due to the nature of the iterative approach. From this table, we can see that our approach returns very good solutions for the majority of these problems. For problems from Model 1, which is very simple, we can always find near-optimal solutions. But when problems become more complex, like those belonging to Model 2 and Model 3, our algorithm does not perform as well. From our analyses and observations, a possible explanation is that the combined heuristic we use takes a greedy approach, which may make bad decisions on scheduling two competitive tasks when a problem is very complex. In particular, when the scheduling time is short, the dual solution may violate many resource capacity constraints so that it is very difficult for the heuristic module to figure out a good feasible solution given such a bad infeasible solution. However, the more CPU time allocated, the less constraints the resulting dual solution will violate. In that case, the heuristic module has a good chance of finding a better primal solution.

Next, we examine the maximum, minimum, mean, and standard deviation of the duality gaps for each group of test cases in Table VII. Clearly, the average duality gap is large in all but the first group of test cases.

When we look at the duality gap for the first set of test cases in Table VIII, we find it is generally small. But the duality gap for the second set in Table IX is generally large. The only difference is that the requested resource number is set to 5 in Table VIII while it is 7 in Table IX. When the requested resource number is uniformly distributed

from 2 to 9, again the duality gap is often very large (Table X). How to interpret this phenomenon? It seems that when the requested resource number is five, two tasks can be satisfied by one supplier at any time, using up *all* the resources of the supplier. However, when the requested resource number is seven, only one task can get enough resources from one supplier at any time, leaving unused resources. What can be deduced from this observation? The Lagrangian relaxation technique transforms the original problem into the relaxed problem, having the following form: $\min[f(x) + \lambda(Ax - b)]$. When resources are underutilized, $\lambda(Ax - b)$ will be negative if the Lagrange multiplier vector λ is not zero. As a result, the dual cost will be less than the primal cost. This explains why the duality gap is big for many test cases. We hypothesize that if resources are nearly fully utilized, the duality gap tends to be small. On the one hand, when there is no competition for resources at all, in one iteration, we can get the optimal solution and the duality gap is zero. On the other hand, when resources are nearly fully utilized at any time having tasks running, the duality gap will also be very small. For cases in between, competition cannot ensure resources to be fully utilized so the duality gap is quite large.

To explore our hypothesis, we tried a small test case containing three tasks, whose information is shown in Table XI. Sup-idx, Res-Num, Pro-Time, Due-Date, and Tardi-Weight are the corresponding supplier index, requested resource number, processing time, due date, and tardiness weight respectively for each task. The optimal solution cost is 210. The best primal solution cost is also 210, but the maximal dual cost is 150. This results in a 40% duality gap. We noticed that after 40 iterations, the dual solution is already feasible ($b_1 = 1, b_2 = 1, b_3 = 5$). But because $\lambda(Ax - b)$ is negative, -61.18, the dual cost is only 148.82. Clearly, the dual cost is a poor lower bound for this small test case.

Another interesting observation is that in case of using the Gaussian distribution to model peak demand, our models still perform well (Table XII). Since the Gaussian distribution cannot lead to a higher resource utilization, the duality gap is big in most cases. Even when the real gaps are above 10% for some problems, it is not bad because these solutions are returned in a short time. With more CPU time allocated, our algorithm can often find better solutions for many problems. Having the ability to produce a solution at any time, our algorithm can be viewed as an anytime algorithm. If we have several such anytime algorithms, they can form a portfolio-based algorithm, which performs well whenever any one of these anytime algorithms can find good solution (Santos et al., 1995; Santos, 2001).

In conclusion, our initial results are very promising, demonstrating the effectiveness of our approach. Although we have not done experiments on Model 4 that take into account those unexpected and dynamic factors, we expect a similar performance on such problems due to the domain independent nature of our relaxation approach.

7. Conclusions and Future Work

We have presented our current work that involved problem formulation and the development of an agent-based model for intra-organizational logistics management. The feasibility of our models is demonstrated through experimental results. Test cases are designed to reflect E-commerce activities with different complexities. Since our problems are NP-hard in general, we especially focused on small problems so that we could compare our results to the optimal solutions obtained via brute force enumeration. From our experiments, we demonstrated that our solutions are typically near optimal, especially when resources are fully utilized by the consumers. Thus, for future work, the idea is to improve the ratio of resource utilization possibly by adding dummy tasks.

In our experiments, we restricted ourselves to a single processor machine in order to more easily observe our algorithm's performance. If distributed multi-agents can be used to solve subproblems in parallel, our computation time will be greatly reduced. We plan to implement our models in a multi-agent environment and measure its performance (Saba and Santos, 2000). Our goal is to solve most problems in less than one minute to meet real world requirements.

The results reported here reflect our initial experiments. We have conducted experiments on larger problems in order to get a better understanding of problem requirements. Ultimately, we need to reformulate our approach even more precisely and test more complex cases, such as those in Model 4. For example, several tasks may be combined together in satisfying a larger goal. To achieve the goal, the related tasks may need to coordinate with each other. These issues are not addressed in the current formulations. In addition, we will try to use distributed nonlinear programming, stochastic programming and other techniques to find optimal or near optimal solution methodologies for these different cases.

Finally, another possible application of this approach is in the air mission planning and execution field. Currently, we are working at large-scale multi-agent distributed goal satisfaction project, which aims at realizing air mission planning and execution in a distributed and dynamic agent environment with a focus on distributed goal satisfaction

(Saba and Santos, 2000). Constructing a separable resource allocation model makes distributed goal satisfaction possible.

Acknowledgements

This work was supported by AFOSR Grant F49620-99-1-0244. A preliminary version of this work can be found in (Santos et al., 2001).

References

Barbuceanu, M. and M. S. Fox: 1996, 'Coordinating Multiple Agents in the Supply Chain'. In: *Proc. 5th Workshops on Enabling Technology for Collaborative Enterprises*. pp. 134–141.

Caldwell, B.: 1995, 'Managing your inventory'. *Information WEEK* **554**.

Chen, H., P. Luh, and L. Fang: 2001, 'A Time Window Based Approach for Job Shop Scheduling'. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*.

Chen, Y., Y. Peng, T. Finin, Y. Labrou, R. Cost, B. Chu, R. Sun, and R. Willhelm: 1999, 'A negotiation-based multi-agent system for supply chain management'. In: *Working Notes of the ACM Autonomous Agents Workshop on Agent-based Decision-support for Managing the Internet-enabled Supply-chain*.

Henoch, J. and H. Ulrich: 2000, 'Agent-Based Management Systems in Logistics'. In: *Proceedings of the ECAI 2000 Workshop 13 "Agent Technologies and Their Application Scenarios in Logistics"*.

Kataoka, N., H. Koizumi, and H. Simizu: 1997, 'Architecture of an autonomous distributed system and verification of implementation as a logistics information management system'. In: *Proc. 3rd Int'l. Workshop on Object-Oriented Real-Time Dependable Systems*.

Kerschberg, L. and S. Banerjee: 1999, 'An agency-based framework for electronic business'. In: *Proceedings of CIA-99 - Third International Workshop on Cooperative Information Agents*.

Liu, J.-S. and K. P. Sycara: 1997, 'Coordination of multiple agents for production management'. *Ann. Ops. Res.* **75**, 235–289.

Luh, P. B., D. Chen, and L. S. Thakur: 1999, 'An Effective Approach for Job-Shop Scheduling with Uncertain Processing Requirements'. *IEEE Trans. on Robot. and Automat.* **15**, 324–335.

Luh, P. B. and D. J. Hoitomt: 1993, 'Scheduling of Manufacturing Systems Using The Lagrangian Relaxation Technique'. *IEEE Trans. on Automat. Control* **38**, 1066–1080.

Modi, P. J., H. Jung, M. Tambe, W.-M. Shen, and S. Kulkarni: 2001, 'A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation'. In: *Proceedings of Seventh International Conference on Principles and Practice of Constraint Programming*. Paphos, Cyprus.

Saba, M. G. and E. Santos, Jr.: 2000, 'The Multi-Agent Distributed Goal Satisfaction System'. In: *Proc. Int'l. ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce*. pp. 389–394.

Sadeh, N. M., D. W. Hildum, D. Kjenstad, and A. Tseng: 1999, 'MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling'. In: *Proc. 3rd Int'l. Conf. on Autonomous Agents Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain*.

Santos, Jr., E.: 2001, 'A Computational Model for Portfolios of Cooperative Heterogeneous Algorithms for Discrete Optimization'. In: *Proceedings of the 14th International FLAIRS Conference*.

Santos, Jr., E., S. E. Shimony, and E. Williams: 1995, 'On a Distributed Anytime Architecture for Probabilistic Reasoning'. Technical report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, AFIT/EN/TR95-02.

Santos, Jr., E., F. Zhang, and P. B. Luh: 2001, 'Multi-Agent Logistics Management'. In: *Proceedings of the 2001 International Conference on Internet Computing (IC'2001)*. pp. 240–246.

Shen, W. and D. H. Norrie: 1998, 'An agent-based approach for manufacturing enterprise integration and supply chain management'. In: G. Jacucci (ed.): *Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise*. Kluwer Academic Publishers, pp. 579–590.

Shen, W., M. Ulieru, D. Norrie, and R. Kremer: 1999, 'Implementing the Internet Enabled Supply Chain through a Collaborative Agent System'. In: *Agents'99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain*.

Swaminathan, J., S. Smith, and N. Sadeh: 1998, 'Modeling Supply Chain Dynamics: A Multiagent Approach'. *Decision Sciences* **29**, 607–632.

Walsh, W. E. and M. P. Wellman: 1998, 'A market protocol for decentralized task allocation'. In: *Proc. 3rd Int'l. Conf. on Multi-Agent Systems*. pp. 325–332.

Walsh, W. E., M. P. Wellman, and F. Ygge: 2000, 'Combinatorial auctions for supply chain formation'. In: *ACM Conference on Electronic Commerce*.

Yung, S. K. and C. C. Yang: 1999, 'A New Approach to Solve Supply Chain Management Problem by Integrating Multi-agent Technology and Constraint Network'. In: *Proc. 32nd Hawaii Int'l. Conf. on System Sciences*.

Zeng, D. D. and K. Sycara: 1999, 'Dynamic Supply Chain Structuring for Electronic Commerce Among Agents'. In: M. Klusch (ed.): *Intelligent Information Agents*. Springer.

Zhao, X., P. B. Luh, and J. Wang: 1999, 'The Surrogate Gradient Method for Lagrangian Relaxation Method'. *J. Optim. Theory and Appl.* **100**, 699–712.

Table I. Parameters for generating test cases

Model	Dist.	S-Num	SN-Per-Task	RT-Num	AR-Num	P-Time	Due-Date	Tardi-Weight	Req-Res-Num
1	Comb-1	3	1	1	10	3∼8	2∼10	{10,20,30}	5
1	Comb-1	3	1	1	10	3∼8	2∼10	{10,20,30}	7
1	Comb-1	3	1	1	10	3∼8	2∼10	{10,20,30}	2∼9
1	Comb-2	3	1	1	1∼10	3∼8	2∼10	{10,20,30}	1∼Res-On-Sup
2	Comb-2	3	1	3	1∼10	3∼8	2∼10	{10,20,30}	1∼Res-On-Sup
3	Comb-2	3	3	3	1∼10	3∼8	2∼10	{10,20,30}	1∼Res-On-Sup
1	Comb-3	3	1	1	1∼10	3∼8	$2\sim10(\sigma=2,\bar{d}=5)$	{10,20,30}	1∼Res-On-Sup
2	Comb-3	3	1	3	1∼10	3∼8	$2\sim10(\sigma=2,\bar{d}=5)$	{10,20,30}	1∼Res-On-Sup
3	Comb-3	3	3	3	1∼10	3∼8	$2\sim10(\sigma=2,\bar{d}=5)$	{10,20,30}	1∼Res-On-Sup

Table II. Real gap percentages by using the original heuristic

Model	Dist.	# of test cases	# of succ. cases	Max	Min	Mean	σ	Comment
1	Comb-1 (Constant)	50	50	8.29	0	0.21	1.18	Req. res. num.=5
1	Comb-1 (Constant)	50	40	19.18	0	5.48	5.27	Req. res. num.=7
1	Comb-1 (Constant)	50	39	27.13	0	5.29	6.85	Req. res. num.=2∼9
1	Comb-2 (Uniform)	50	46	55.26	0	4.50	10.42	
2	Comb-2 (Uniform)	50	39	44.38	0	5.38	8.43	
3	Comb-2 (Uniform)	50	35	30.19	0	7.35	8.48	
1	Comb-3 (Gaussian)	50	46	48.48	0	3.20	8.38	
2	Comb-3 (Gaussian)	50	45	32.89	0	2.70	6.84	
3	Comb-3 (Gaussian)	50	37	29.06	0	5.89	6.83	

Table III. Real gap percentages by using the WSPT/CR heuristic

Model	Dist.	# of test cases	# of succ. cases	Max	Min	Mean	σ	Comment
1	Comb-1 (Constant)	50	50	5.83	0	0.15	0.84	Req. res. num.=5
1	Comb-1 (Constant)	50	50	4.04	0	0.09	0.57	Req. res. num.=7
1	Comb-1 (Constant)	50	49	73.45	0	1.71	10.33	Req. res. num.=2∼9
1	Comb-2 (Uniform)	50	48	44.44	0	1.20	6.43	
2	Comb-2 (Uniform)	50	47	54.90	0	2.12	8.65	
3	Comb-2 (Uniform)	50	41	27.68	0	4.37	7.08	
1	Comb-3 (Gaussian)	50	50	8.75	0	0.38	1.48	
2	Comb-3 (Gaussian)	50	48	21.46	0	0.72	3.54	
3	Comb-3 (Gaussian)	50	44	21.78	0	2.69	4.89	

Table IV. Some test cases where the WSPT/CR heuristic performs worse than the original heuristic.

Model	Dist.	Case-No	Ori-Heu	WSPT-Heu	Optimal-Cost	Comment
1	Comb-1	24	2400.00	2540.00	2400.00	Req. res. num.=5
1	Comb-1	49	2400.00	3920.00	2260.00	Req. res. num.=5
2	Comb-2	15	2510	2680.00	2510.00	
2	Comb-2	30	3060.00	4740.00	3060.00	
2	Comb-2	47	17770.00	22720.00	17770.00	
2	Comb-2	49	9210.00	10200.00	9210.00	
3	Comb-2	10	67393.00	72477.00	67321.00	
3	Comb-2	17	21531.00	24492.00	21531.00	
3	Comb-2	27	19433.00	22775.00	18622.00	
2	Comb-2	38	53133.00	58610.00	53133.00	
1	Comb-3	13	7080	7430.00	7080.00	
3	Comb-3	25	42256.00	44828.00	39418.00	
3	Comb-3	27	35823.00	37631.00	34493.00	
3	Comb-3	30	50830.00	52012.00	50830.00	
3	Comb-3	41	102942.00	107622.00	98790.00	
3	Comb-3	43	28573.00	32592.00	26764.00	

Table V. Real gap percentages by using the combined heuristics

Model	Dist.	# of test cases	# of succ. cases	Max	Min	Mean	σ	Comment
1	Comb-1 (Constant)	50	50	1.45	0	0.03	0.20	Req. res. num.=5
1	Comb-1 (Constant)	50	50	0.47	0	0.02	0.08	Req. res. num.=7
1	Comb-1 (Constant)	50	50	8.79	0	0.29	1.42	Req. res. num.=2∼9
1	Comb-2 (Uniform)	50	50	6.81	0	0.17	0.96	
2	Comb-2 (Uniform)	50	50	2.75	0	0.11	0.44	
3	Comb-2 (Uniform)	50	45	27.68	0	3.29	5.87	
1	Comb-3 (Gaussian)	50	50	8.75	0	0.29	1.33	
2	Comb-3 (Gaussian)	50	47	21.46	0	0.97	3.94	
3	Comb-3 (Gaussian)	50	42	20.51	0	3.09	5.01	

Table VI. Test cases where the combined heuristic performs worse than either the original heuristic or the WSPT/CR heuristic. If the primal cost obtained by the combined heuristic is greater than those given by both the original heuristic and the WSPT/CR heuristic, it is shown in bold face.

Model	Dist.	Case-No	Ori-Heu	WSPT-Heu	Comb-Heu	Opt-Cost
3	Comb-2	6	76935.00	77121.00	77085.00	73997.00
3	Comb-2	12	83898.00	83696.00	83898.00	71150.00
3	Comb-2	18	91885.00	79650.00	81730.00	78252.00
3	Comb-2	22	58369.00	56929.00	58369.00	56909.00
3	Comb-2	28	25666.00	23635.00	25666.00	23624.00
3	Comb-2	30	80427.00	79591.00	79610.00	72200.00
3	Comb-2	31	17690.00	17500.00	17512.00	17500.00
3	Comb-2	46	104065.00	91215.00	92815.00	91215.00
2	Comb-3	36	6760.00	6760.00	**7640.00**	6760.00
3	Comb-3	1	56691.00	55261.00	55341.00	55261.00
3	Comb-3	5	87681.00	87685.00	**88395.00**	79648.00
3	Comb-3	11	47232.00	47274.00	47274.00	47224.00
3	Comb-3	20	37140.00	34369.00	36747.00	34369.00
3	Comb-3	21	52021.00	52021.00	**52139.00**	46798.00
3	Comb-3	22	73555.00	57418.00	68682.00	56995.00
3	Comb-3	24	91391.00	85691.00	86271.00	85681.00
3	Comb-3	25	42256.00	44828.00	44828.00	39418.00
3	Comb-3	27	35823.00	37631.00	**38144.00**	34493.00
3	Comb-3	29	42133.00	34067.00	34083.00	34067.00
3	Comb-3	30	50830.00	52012.00	50842.00	50830.00
3	Comb-3	37	37563.00	35464.00	35556.00	34216.00
3	Comb-3	38	37175.00	33487.00	37175.00	32625.00
3	Comb-3	41	102942.00	107622.00	107622.00	98790.00
3	Comb-3	45	20559.00	20553.00	20559.00	20553.00

Table VII. Duality gap percentages

Model	Dist.	# of test cases	Max	Min	Mean	σ	Comment
1	Comb-1	50	14.36	0	2.05	3.18	Req. res. num.=5
1	Comb-1	50	241.42	108.37	158.55	33.03	Req. res. num.=7
1	Comb-1	50	304.09	0.78	87.83	62.36	Req. res. num.=2∼9
1	Comb-2	50	260.04	11.59	86.63	62.13	
2	Comb-2	50	439.84	0	115.86	80.76	
3	Comb-2	50	124.39	18.74	59.93	23.77	
1	Comb-3	50	317.24	0	55.55	55.31	
2	Comb-3	50	206.05	7.32	76.67	48.12	
3	Comb-3	50	100.84	12.59	53.63	22.03	

Table VIII. Test cases of Model 1 based on Comb-1 with the requested resource number = 5, $mean_{gap} = 0.03, \sigma_{gap} = 0.20$, 50/50 are within 10% of the optimal cost

Case-No	Primal-Cost	Dual-Cost	Duality-Gap	Optimal-Cost	Real-Gap
1	5440.00	5401.80	0.71	5440.00	0.00
2	2310.00	2309.92	0.00	2310.00	0.00
3	290.00	277.61	4.46	290.00	0.00
4	1990.00	1989.93	0.00	1990.00	0.00
5	2720.00	2716.29	0.14	2720.00	0.00
6	3170.00	2910.74	8.91	3170.00	0.00
7	9140.00	9082.40	0.63	9140.00	0.00
8	1280.00	1279.80	0.02	1280.00	0.00
9	1660.00	1658.24	0.11	1660.00	0.00
10	780.00	763.61	2.15	780.00	0.00
11	7400.00	6890.72	7.39	7400.00	0.00
12	1620.00	1618.30	0.11	1620.00	0.00
13	3410.00	3303.10	3.24	3410.00	0.00
14	2870.00	2739.13	4.78	2870.00	0.00
15	4060.00	4004.38	1.39	4060.00	0.00
16	5660.00	5660.00	0	5660.00	0.00
17	4160.00	4120.55	0.96	4160.00	0.00
18	1980.00	1979.94	0.00	1980.00	0.00
19	850.00	825.32	2.99	850.00	0.00
20	3440.00	3425.39	0.43	3440.00	0.00
21	4280.00	4252.57	0.65	4280.00	0.00
22	5070.00	5036.48	0.67	5070.00	0.00
23	3910.00	3882.07	0.72	3910.00	0.00
24	2400.00	2399.97	0.00	2400.00	0.00
25	1580.00	1579.98	0.00	1580.00	0.00
26	1370.00	1363.39	0.48	1370.00	0.00
27	1510.00	1509.96	0.00	1510.00	0.00
28	940.00	932.43	0.81	940.00	0.00
29	1560.00	1560.00	0.00	1560.00	0.00
30	15230.00	15059.58	1.13	15230.00	0.00
31	1470.00	1458.15	0.81	1470.00	0.00
32	3840.00	3840.00	0.00	3840.00	0.00
33	5170.00	4647.79	11.24	5170.00	0.00
34	6340.00	6289.29	0.81	6340.00	0.00
35	3070.00	2813.08	9.13	3070.00	0.00
36	1030.00	1029.81	0.02	1030.00	0.00
37	5080.00	4841.68	4.92	5080.00	0.00
38	3540.00	3528.58	0.32	3540.00	0.00
39	480.00	479.99	0.00	480.00	0.00
40	3740.00	3667.62	1.97	3740.00	0.00
41	3180.00	3170.40	0.30	3180.00	0.00
42	7580.00	7412.42	2.26	7580.00	0.00
43	2830.00	2739.43	3.31	2830.00	0.00
44	2410.00	2409.08	0.04	2410.00	0.00
45	6310.00	6168.75	2.29	6220.00	1.45
46	2530.00	2212.31	14.36	2530.00	0.00
47	100.00	100.00	0	100.00	0.00
48	4590.00	4323.23	6.17	4590.00	0.00
49	7600.00	7599.98	0.00	7600.00	0.00
50	190.00	186.77	1.73	190.00	0.00

Table IX. Test cases of Model 1 based on Comb-1 with the requested resource number = 7, $mean_{gap} = 0.02, \sigma_{gap} = 0.08$, 50/50 are within 10% of the optimal cost

Case-No	Primal-Cost	Dual-Cost	Duality-Gap	Optimal-Cost	Real-Gap
1	24800.00	10183.02	143.54	24800.00	0
2	30240.00	14363.00	110.54	30240.00	0
3	5470.00	1973.96	177.11	5470.00	0
4	24530.00	10399.97	135.87	24530.00	0
5	13040.00	3929.55	218.87	12530.00	0
6	15340.00	6379.24	140.47	15340.00	0
7	22030.00	9542.83	130.85	22030.00	0
8	6570.00	2511.07	211.14	6570.00	0
9	66250.00	31382.66	109.38	66250.00	0
10	14320.00	6424.96	128.59	14320.00	0
11	16420.00	6630.35	151.99	16420.00	0
12	12640.00	3960.28	219.96	12640.00	0
13	5310.00	1600.63	247.93	5310.00	0
14	20570.00	8449.73	171.09	20570.00	0
15	39160.00	14842.55	159.26	39160.00	0
16	13610.00	6247.30	117.85	13610.00	0
17	8120.00	2914.21	178.63	8120.00	0
18	31090.00	11557.83	169.00	31090.00	0
19	26470.00	10011.14	164.41	26380.00	0.34
20	14720.00	6210.27	137.03	14720.00	0
21	10570.00	4269.11	147.59	10570.00	0
22	22790.00	10937.10	108.37	22790.00	0
23	19390.00	7977.59	143.06	19300.00	0.47
24	17250.00	7761.19	122.26	17250.00	0
25	10730.00	3597.42	198.27	10730.00	0
26	15820.00	5335.00	196.53	15820.00	0
27	31730.00	12102.01	162.19	31730.00	0
28	6800.00	2849.71	138.62	6800.00	0
29	46090.00	19846.19	132.24	46090.00	0
30	16940.00	7458.91	127.11	16940.00	0
31	18810.00	7046.72	166.93	18810.00	0
32	24300.00	8884.26	173.52	24300.00	0
33	8150.00	2683.81	203.67	8150.00	0
34	10590.00	4136.55	156.01	10590.00	0
35	41450.00	16470.06	151.67	41450.00	0
36	19630.00	8613.05	127.91	19630.00	0
37	18850.00	8987.67	109.73	18850.00	0
38	24840.00	9124.51	172.23	24840.00	0
39	4830.00	1414.70	241.42	4830.00	0
40	34440.00	13521.68	154.70	34440.00	0
41	10870.00	3802.56	185.86	10870.00	0
42	11740.00	4021.26	191.95	11740.00	0
43	8550.00	2693.13	217.47	8550.00	0
44	25320.00	10669.67	137.31	25320.00	0
45	31530.00	12974.67	143.01	31530.00	0
46	28210.00	11382.81	147.83	28210.00	0
47	26170.00	9292.58	181.62	26170.00	0
48	18360.00	7643.90	140.19	18360.00	0
49	23710.00	10211.65	132.19	23710.00	0
50	19560.00	6731.23	190.59	19560.00	0

Table X. Test cases of Model 1 based on Comb-1 with the requested resource number from 2 to 9, $mean_{gap} = 0.29, \sigma_{gap} = 1.42$, 50/50 are within 10% of the optimal cost

Case-No	Primal-Cost	Dual-Cost	Duality-Gap	Optimal-Cost	Real-Gap
1	11980.00	5252.59	128.08	11980.00	0.00
2	920.00	912.87	0.78	920.00	0.00
3	3390.00	1878.72	80.44	3390.00	0.00
4	4740.00	2150.39	120.43	4740.00	0.00
5	8110.00	3447.78	135.22	8110.00	0.00
6	9680.00	4027.93	140.32	9620.00	0.62
7	5710.00	4187.46	36.36	5710.00	0.00
8	18270.00	11795.73	54.89	18270.00	0.00
9	28880.00	12056.45	139.54	28880.00	0.00
10	4210.00	3144.93	33.87	4210.00	0.00
11	470.00	390.00	20.51	470.00	0.00
12	2180.00	754.35	188.99	2180.00	0.00
13	3140.00	777.06	304.09	3140.00	0.00
14	4630.00	3122.92	48.26	4630.00	0.00
15	960.00	813.59	18.00	960.00	0.00
16	2660.00	1757.58	51.34	2660.00	0.00
17	13740.00	6202.92	121.51	13740.00	0.00
18	43690.00	20937.43	108.67	40160.00	8.79
19	5420.00	4228.83	28.17	5420.00	0.00
20	8080.00	6955.94	16.16	8080.00	0.00
21	10820.00	5846.00	85.08	10820.00	0.00
22	2400.00	1899.23	27.76	2400.00	0.00
23	12140.00	4877.06	148.92	12140.00	0.00
24	14960.00	10733.34	39.38	14960.00	0.00
25	9050.00	4788.95	88.98	9050.00	0.00
26	14480.00	7851.89	84.41	14480.00	0.00
27	9650.00	5985.31	61.23	9650.00	0.00
28	1890.00	998.45	89.29	1890.00	0.00
29	4460.00	4177.75	6.76	4460.00	0.00
30	3100.00	2703.80	14.65	3100.00	0.00
31	1250.00	1033.66	20.93	1250.00	0.00
32	6390.00	4048.82	57.82	6390.00	0.00
33	640.00	2499.53	156.05	6400.00	0.00
34	3670.00	2026.99	81.06	3670.00	0.00
35	9940.00	5410.50	83.72	9940.00	0.00
36	24830.00	17041.23	45.71	24830.00	0.00
37	9050.00	3202.02	182.63	9050.00	0.00
38	9320.00	3927.51	137.30	9320.00	0.00
39	7650.00	4114.03	85.95	7650.00	0.00
40	8270.00	4339.89	90.56	8270.00	0.00
41	8720.00	2269.97	284.15	8720.00	0.00
42	15640.00	8988.02	74.01	14860.00	5.25
43	8910.00	5161.39	72.63	8910.00	0.00
44	5540.00	2577.77	114.91	5540.00	0.00
45	54250.00	33013.32	64.33	54250.00	0.00
46	15630.00	8749.98	78.63	15630.00	0.00
47	5130.00	2387.50	114.87	5130.00	0.00
48	830.00	567.53	46.25	830.00	0.00
49	2260.00	1417.64	59.42	2260.00	0.00
50	7440.00	3402.05	118.69	7440.00	0.00

Table XI. Description of the three-job example

Task-No	Sup-Idx	Res-Num	Pro-Time	Due-Date	Tardi-Weight
1	2	8	4	3	20
2	1	5	3	2	10
3	2	6	4	5	20

Table XII. Test cases of Model 3 based on the Gaussian distribution, $mean_{gap} = 3.09, \sigma_{gap} = 5.01$, 42/50 are within 10% of the optimal cost

Case-No	Primal-Cost	Dual-Cost	Duality-Gap	Optimal-Cost	Real-Gap
1	55341.00	34209.03	61.77	55261.00	0.14
2	37073.00	20566.83	80.26	37013.00	0.16
3	23673.00	14603.03	62.11	20896.00	13.29
4	37976.00	29082.30	30.58	37976.00	0.00
5	88395.00	48315.01	82.96	79648.00	10.98
6	19749.00	16459.44	19.99	19749.00	0.00
7	44096.00	27431.21	60.75	44096.00	0.00
8	14581.00	12950.57	12.59	14581.00	0.00
9	38786.00	24629.03	57.48	37793.00	2.63
10	58354.00	33101.44	76.29	58351.00	0.01
11	47274.00	36297.31	30.24	47224.00	0.11
12	51709.00	25746.24	100.84	51709.00	0.00
13	22274.00	12572.67	77.16	22262.00	0.05
14	95348.00	53614.23	77.84	95326	0.02
15	20611.00	14939.85	37.96	20611.00	0.00
16	24835.00	17014.89	45.96	24827.00	0.03
17	32563.00	24611.36	32.31	32443.00	0.37
18	24834.00	16601.48	49.59	23948.00	3.70
19	17527.00	9867.81	77.62	17517.00	0.06
20	36747.00	24984.36	47.08	34369.00	6.92
21	52139.00	29745.48	75.28	46798.00	11.41
22	68682.00	39383.05	74.39	56995.00	20.51
23	33612.00	21859.80	53.76	33612.00	0.00
24	86271.00	43478.68	98.42	85681.00	0.69
25	44828.00	25298.25	77.20	39418.00	13.72
26	26262.00	17710.44	48.29	26163.00	0.38
27	38144.00	25891.15	47.32	34493.00	10.58
28	47458.00	30636.55	54.91	46828.00	1.35
29	34083.00	25788.32	32.16	34067.00	0.05
30	50842.00	31441.90	61.70	50830.00	0.02
31	19925.00	15811.56	26.02	19925.00	0.00
32	13079.00	10534.63	24.15	13079.00	0.00
33	10239.00	7478.43	36.91	10238.00	0.01
34	27040.00	20696.28	30.65	26582.00	1.72
35	30570.00	22806.38	34.04	30567.00	0.01
36	24645.00	14601.07	68.79	22978.00	7.25
37	35556.00	27443.70	29.56	34216.00	3.92
38	37175.00	23103.58	60.91	32625.00	13.95
39	22656.00	14956.76	51.48	22656.00	0.00
40	43793.00	23371.09	87.38	39616.00	10.54
41	107622.00	58505.63	83.95	98790.00	8.94
42	50196.00	35924.28	39.73	50177.00	0.04
43	28573.00	16714.55	70.95	26764.00	6.76
44	38781.00	28777.81	34.76	38760.00	0.05
45	20559.00	17206.27	19.49	20553.00	0.03
46	78358.00	63095.86	24.19	77618.00	0.95
47	35166.00	23946.00	46.86	34165.00	2.93
48	32556.00	21956.81	48.27	32540.00	0.05
49	51784.00	36091.33	43.48	51772.00	0.02
50	95466.00	54475.29	75.25	95466.00	0.00