Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 35 (2008) 136–142
Expert Systems with Applications
www.elsevier.com/locate/eswa

Multi-objective inventory planning using MOPSO and TOPSIS

Ching-Shih Tsou *

Department of Business Administration, National Taipei College of Business, Taipei 10051, Taiwan

Abstract

One of the main characteristics of today's business tends to vary often. Under such environment, many decisions should be carefully pondered over from relevant aspects which are usually conflicting. Hence, inventory planning problems, which address how much and when to order what customers need at the least relevant cost while maintaining a desirable service level expected by customers, could be recast into a multi-objective optimization problem (MOOP). In a MOOP there are normally infinite numbers of optimal solutions in the Pareto front due to the conflicts among objectives. Unfortunately, most multi-objective inventory models have been solved by aggregation methods through a linear combination of specific weights or only one objective was optimized and the others were turned into constraints. Therefore, the challenges decision makers face are not only modeling the problem in a multi-objective context, but also the effort dedicated to build the Pareto front of MOOPs. This paper first employs the multi-objective particle swarm optimization (MOPSO) algorithm to generate the non-dominated solutions of a reorder point and order size system. A ranking method called technique for order preference by similarity to ideal solution (TOPSIS) is then used to sort the non-dominated solutions by the preference of decision makers. That is, a two-stage multi-criteria decision framework which consists of MOPSO and TOPSIS is presented to find out a compromise solution for decision makers. By varying the weights of various criteria, including minimization of the annual expected total relevant cost, minimization of the annual expected frequency of stock-out occasions, and minimization of the annual expected number of stock-outs, managers can determine the order size and safety stock simultaneously which fits their preference under different situations.

Keywords: Particle swarm optimization; Multi-objective optimization; Inventory planning

1. Introduction

Most real-world problems in business are modeled as an optimization problem involving a single objective. The assumption that firms always seek to maximize (or minimize) their profit (or cost) rather than making tradeoffs among multiple objectives has been criticized for a long time. For example, an inventory control system should operate at least cost while maintaining desirable service level expected by customers. The objectives of cost minimization and service level maximization are incommensurate and conflicting with each other. Hence, the stage is set for a multi-objective analysis of decision making not only in business, but also for engineering design and scientific experiments.

Up to now, most multi-objective inventory models have put their emphasis on deteriorating items. Padmanabhan and Vrat (1990) solved a multi-objective inventory of deteriorating items with stock-dependent demand by a nonlinear goal programming method. Agrell (1995) presented a decision support system for multi-criteria inventory control. The solution procedure embedded is an interactive method with preferences extracted progressively in decision analysis process to determine batch size and security stock. Roy and Maiti (1998) formulated a multi-objective inventory model of deteriorating items with stock-dependent demand under limited imprecise storage area and total cost budget. The objectives therein are to maximize the profit and to minimize the wastage cost where the profit goal, wastage cost and storage area are fuzzy in nature. The

* Tel.: +886 2 23226459; fax: +886 2 27942285.
E-mail address: cstsou@mail.ntcb.edu.tw

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.06.009

problem was solved by Fuzzy Non-Linear Programming (FNLP) and Fuzzy Additive Goal Programming (FAGP). Mahapatra and Maiti (2005) considered a multi-objective inventory model of stochastically deteriorating items and incorporated the impact of quality level into the demand and deterioration function. Mandal, Roy, and Maiti (2005) presented a multi-item multi-objective fuzzy inventory model with three constraints, actually a cost minimization objective with three fuzzy goal constraints, to find the demand, order size, and shortage level for each item. They solved the problem by geometric programming method.

All works mentioned above using a relative weight vector to scalarize multiple objectives into a single objective and solve the corresponding problem by traditional single-objective optimization techniques. This approach contradicts our intuition that single-objective optimization is a degenerate case of MOOP (Deb, 2001). Furthermore, a MOOP does not have a single solution that could optimize all objectives simultaneously. Therefore, solving MOOP is not to search for optimal solutions but for efficient solutions that can be expressed in terms of non-dominated solutions in the objective space. A solution is said to be dominant over another only if it has superior, at least no inferior, performance in all objectives. The result of preference-based approach is a compromise solution whose non-dominance is not guaranteed (Liu, Yang, & Whidborne, 2003). Lastly, but not the least, a single optimized solution could only be found in each simulation run of traditional optimization techniques. Therefore, using a population of solutions to evolve towards several non-dominated solutions in each run makes evolutionary algorithms (EA) popular in solving MOOPs.

PSO is also a population-based stochastic optimization heuristic developed by Kennedy and Eberhart (1995), which was inspired by social behavior of bird flocking or fish schooling. It has been an effective technique to search for optima of optimization problems. Although it does not use the selection operation because the members of the entire population are maintained throughout the search procedure, PSO actually shares many similarities with evolutionary computation techniques (Eberhart & Shi, 1998). Applications, parameter selection, and modified versions of PSO can be found in Eberhart and Shi (2001) and Shi and Eberhart (1998a, 1998b).

One of the successful applications of PSO to MOOPs, named multi-objective PSO (MOPSO), is the seminal work of Coello-Coello and Lechuga (2002). In a subsequent study done by them, MOPSO is not only a viable alternative to solve MOOPs, but also the only one, compared with the non-dominated sorting genetic algorithm-II (NSGA-II) (Deb, Pratap, Agarwal, & Meyarivan, 2002), the Pareto archive evolutionary strategy (PAES) (Knowles & Corne, 2000), and the micro-genetic algorithm (microGA) (Coello-Coello & Pulido, 2001) for multi-objective optimization, can cover the full Pareto front MOPSO of all the test functions therein (Coello-Coello, Pulido, & Lechuga, 2004).

In addition to generating possible tradeoff solutions by considering all objectives, the effort devoted to solicit a compromise solution amenable to decision maker's preference is equally important. An ideal approach to analyze the decision in a multi-objective context is first to find multiple tradeoff optimal solutions, so called non-dominated solutions, of a MOOP. Then, one of the obtained solutions is chosen by using preference information of decision makers. It is normally implemented as a ranking process. In a nutshell, resolution of MOOP involves two stages, one is how to find the non-dominated front in the objective space, and the other is how to rank the non-dominated solutions by subjective judgments or preference information provided by decision makers. Both stages require an iterative solution procedure in which the decision maker investigates a variety of solutions to find one or some that is most satisfactory. Therefore, this paper first applies the MOPSO to generate the non-dominated solutions of order size and safety stock in a multi-objective inventory planning model. After that, a ranking method called technique for order preference by similarity to ideal solution (TOPSIS) is used to prioritize the non-dominated solutions for decision makers (Yoon & Hwang, 1995).

The rest of this paper is organized as follows. Section 2 reviews a multi-objective inventory planning model presented by Agrell (1995). Next, some preliminary knowledge about PSO and MOOP are described in Section 3. Sections 4 and 5 report the solution procedure and the computational results, respectively. Finally, conclusions and future research directions are drawn out in Section 6.

2. Multi-objective inventory planning

The control of inventories has been a major issue in the field of industrial engineering and operational research (IE/OR) for a long time. As an essential activity for any enterprise, inventory planning tries to determine the decisions about when to order and how much should order for different control mechanisms. A common control mechanism is a continuous-review (r,Q) system in which an order of size Q is placed whenever the inventory position drops to the reorder point, r (Silver, Pyke, & Peterson, 1998). Determination of (r, Q) depends on the lead time and the fluctuation of demand which aims to minimize inventory cost and maximize customer service. Agrell (1995) set up three objectives about cost and stock-out to plan for two control parameters- the order size Q and the safety factor k which is a term of the reorder point r. It is assumed that only a single product is considered here. After a fixed lead time, L, the order is received all at once and placed into inventory. The demand within lead time is a random variable having a normal probability distribution function $f_{D_L}(x)$ with mean μ_L and standard deviation σ_L. The system is also characterized by the following parameters.

D	average annual demand
A	ordering cost

c	unit item cost
h	inventory carrying rate
k	safety factor

The multi-objective inventory planning model is

$$\underset{Q,K}{\text{Minimize}}\, C(Q,k) = \frac{AD}{Q} + hc\left(\frac{Q}{2} + k\sigma_L\right) \tag{1}$$

$$\underset{Q,K}{\text{Minimize}}\, N(Q,k) = \frac{D}{Q}\int_r^{\infty} f_{D_L}(x)\mathrm{d}x \tag{2}$$

$$\underset{Q,K}{\text{Minimize}}\, S(Q,k) = \frac{D}{Q}\int_r^{\infty} (x-r) f_{D_L}(x)\mathrm{d}x \tag{3}$$

Subject to

$$0 \leqslant Q \leqslant D \tag{4}$$

$$0 \leqslant k \leqslant D/\sigma_L \tag{5}$$

Eq. (1) is to minimize the expected total relevant cost annually. The first term gives the order cost per cycle multiplied by the number of annual cycles. The second term is the holding cost, which amounts to half the order quantity plus the safety stock. Eq. (2) is to minimize the expected frequency of stock-out occasions annually, and Eq. (3) is to minimize the expected number of items stocked out annually. Eq. (4) ensures that the order size must be non-negative and no greater than the average annual demand. Eq. (5) guarantees that the safety stock should be non-negative and no greater than the average annual demand.

3. Pareto optimality concepts

Pareto optimality is the most important solution concept in MOOP. It is also a major way to find the equilibrium results in non-cooperative game theory. Before presenting the approach to plan for the control parameters in a multi-objective inventory planning context, some preliminary knowledge about Pareto optimality is briefly introduced in this section.

A MOOP problem with K objectives and M constraints can be stated as follows.

$$\text{Minimize} \quad \vec{\mathbf{f}}(\vec{\mathbf{x}}) = [f_1(\vec{\mathbf{x}}), f_2(\vec{\mathbf{x}}), \ldots, f_K(\vec{\mathbf{x}})]^T \tag{6}$$

$$\text{Subject to} \quad \vec{\mathbf{x}} \in \Omega \tag{7}$$

$$\Omega = \{\vec{\mathbf{x}} | g_j(\vec{\mathbf{x}}) \leqslant 0, \quad j = 1,2,\ldots,M\}, \tag{8}$$

where $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_D]^T$ is a D dimensional vector, each $x_i (i = 1,2,\ldots,D)$ can be real-valued, integer-valued or boolean-valued. Functions $f_i(\vec{\mathbf{x}})$ $(i = 1,2,\ldots,K)$ and $g_j(\vec{\mathbf{x}}) (j = 1,2,\ldots,M)$ could be linear or nonlinear arbitrary functions. The MOOP is to find the vector $\vec{\mathbf{x}}^* = [x_1^*, x_2^*, \ldots, x_D^*]^T$ which will satisfy the M constraints and minimize the vector objective function $\vec{\mathbf{f}}(\vec{\mathbf{x}}) = [f_1(\vec{\mathbf{x}}), f_2(\vec{\mathbf{x}}), \ldots, f_K(\vec{\mathbf{x}})]^T$.

Solving a MOOP problem is to find the non-dominated solutions in the objective space and its counterparts in the decision space which are called as the efficient solutions. We do not intend to strictly differentiate these two terms in the following discussion. In a single-objective optimization problem, any two solutions can be compared completely. However, because of the conflicting objectives in a MOOP, not all solutions could be compared completely. So we need to clarify the possible partial orderings of the objective vectors which relate to the Pareto dominance concept. For a minimization MOOP problem, a decision vector $\vec{\mathbf{u}} = (u_1, u_2, \ldots, u_D)$ is said to strongly dominate $\vec{\mathbf{v}} = (v_1, v_2, \ldots, v_D)$ (denoted by $\vec{\mathbf{u}} \prec \vec{\mathbf{v}}$) if and only if $\forall i \in \{1,2,\ldots,K\}$, $f_i(\vec{\mathbf{u}}) < f_i(\vec{\mathbf{v}})$. Less stringently, a decision vector $\vec{\mathbf{u}}$ weakly dominate $\vec{\mathbf{v}}$ (denoted by $\vec{\mathbf{u}} \preceq \vec{\mathbf{v}}$) if and only if $\forall i \in \{1,2,\ldots,K\}$, $f_i(\vec{\mathbf{u}}) \leqslant f_i(\vec{\mathbf{v}})$ and $\exists i \in \{1,2,\ldots,K\}$, $f_i(\vec{\mathbf{u}}) < f_i(\vec{\mathbf{v}})$. A solution is dominant over another only if it has superior performance in all criteria. A set of decision vectors is said to be a non-dominated set if no member of the set is dominated by any other member. Hence, non-dominance means that the improvement of one objective could only be achieved at the expense of other objectives. The collection of all efficient solutions of a minimization MOOP is called the efficient set. The image of the efficient set by $\vec{\mathbf{f}}$ is referred to as the Pareto front. That is, the Pareto front, $\widetilde{P}$, is the non-dominated set of solutions which are not dominated by any feasible solution. And there are normally infinite numbers of non-dominated solutions due to the conflicts among objectives.

4. A multi-criteria decision framework

A two-stage multi-criteria decision framework of the inventory planning problem is presented in Fig. 1. In the first stage, decision makers set up the parameters needed in MOPSO to generate the non-dominated front of the multi-objective inventory planning problem. Next, according to the weights reflecting the preference on each objective, TOPSIS is used to rank the non-dominated

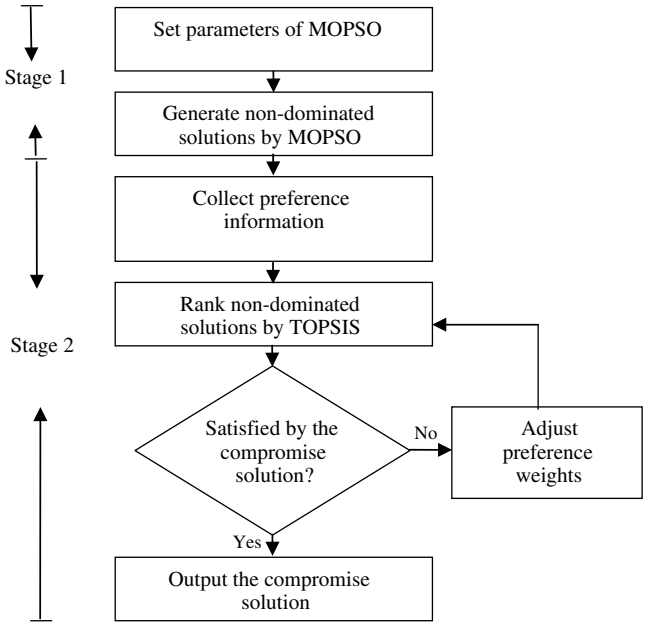

Fig. 1. A two-stage multi-criteria decision framework for inventory planning.

solutions generated in the first stage. Decision makers could adjust their preference to find the compromise solution that they satisfy most. We will discuss the MOPSO in the first subsection and what comes after is the TOPSIS.

4.1. Multi-objective particle swarm optimization

To speak of MOPSO, let us start with the PSO. In PSO, the system is initialized with a population of random solutions, called "particles". All particles have fitness values that are evaluated by the function to be optimized. Each particle flies through the problem space with a velocity, which is constantly updated by the particle's own experience and the best experience of its neighbors, in order to search for optima by iterations. Compared to GA, the advantages of PSO are that it is easy to implement and there are fewer parameters to adjust.

In every iteration, each particle is updated by following two best values. The first one is the best solution it has achieved so far. This value is called *pbest*. Another "best" value that is tracked by the optimizer is the best value obtained so far by all particles in its neighborhood. This best value is a local best and called *lbest*. If the neighborhood is defined as the whole population, each particle will move towards its best previous position and towards the best position ever been in the whole swarm, called *gbest* model.

In this paper, we use the global version of PSO. After finding the two best values, the velocity and position of each particle are updated by the following equations.

$$v_{nd}^{\text{new}} = \omega \cdot v_{nd}^{\text{old}} + c_1 \cdot \text{RAND} \cdot (p_{nd} - x_{nd}) + c_2 \cdot \text{RAND} \cdot (g_d - x_{nd}) \quad (9)$$

$$x_{nd}^{\text{new}} = x_{nd} + v_{nd}^{\text{new}} \quad (10)$$

where

v_{nd}^{old} is the old velocity of particle n along dimension d,
v_{nd}^{new} is the new velocity of particle n along dimension d,
ω is the inertia weight which is between 0.8 and 1.2,
c_1 and c_2 are the learning factors (or acceleration coefficients), usually between 1 and 4,
x_{nd} is the current position of particle n along dimension d,
p_{nd} is the individual best solution of particle n along dimension d,
g_d is the global best solution the whole population ever been along dimension d, and
RAND is a random numbers between 0 and 1.

In general, the pseudo-code for PSO is as follows:

Initialize population
Evaluate fitness
Do while the maximum number of iterations has not been reached
Find individual best
Find global best
Update velocity according to Eq. (9)
Update position according to Eq. (10)
Evaluate fitness
Loop

Note that particles' velocities on each dimension are restricted to a maximum velocity. If a new velocity along certain dimension exceeds the maximum velocity specified by the user, then the velocity along that dimension is limited to its maximum velocity. Finally, the criterion used to stop the search process is whether a maximum number of iterations have been reached.

The difficulty in extending the PSO to MOOPs is how to select a global guide for each particle. In MOOPs, because there is not a single optimum, the non-dominated solutions found by MOPSO so far are all stored in an archive. Each particle can randomly select a non-dominated solution from the archive as the global guide for its next flight. Although this selection method is simple, it can promote convergence (Alvarez-Benitez, Everson, & Fieldsend, 2005). The pseudo-code of MOPSO is shown in Table 1.

To run the MOPSO, we need to specify the number of iterations (T) and the number of particles (N) first. After initializing each particle's position ($\vec{\mathbf{x}}_n^D$) and velocity ($\vec{\mathbf{v}}_n^D$), the individual and global best solutions for particle n is set to be its initial position. That is, $\vec{\mathbf{p}}_n^D = \vec{\mathbf{x}}_n^D$ and $\vec{\mathbf{g}}_n^D = \vec{\mathbf{x}}_n^D$. At each iteration, we randomly choose a solution from the non-dominated archive, $\widetilde{A}$, as a global best solution for each particle. Then the velocity and position of each particle are updated according to Lines 07 and 08. Notice that each particle could have a different global guide for its each flight. Besides that, a possibly different global guide is associated with each particle. This is why the global best solutions carry a subscript n only for MOPSO.

Table 1
The pseudo-code of MOPSO

```
01    Ã = ∅
02    {x⃗_n^D, v⃗_n^D, p⃗_n^D}_{n=1}^N = Initialize()
03    for t = 1 to T
04      for n = 1 to N
05      g⃗_n^D = Random(Ã)
06        for d = 1 to D
07          v_nd^new = ω · v_nd^old + c1 · r1 · (p_nd − x_nd) + c2 · r2 · (g_nd − x_nd)
08          x_nd = x_nd + v_nd^new
09        end for
10        y⃗_n^K = f⃗(x⃗_n^D) = [f1(x⃗_n^D), f2(x⃗_n^D), ..., fK(x⃗_n^D)]
11        if x⃗_n^D ⊁ u⃗^D, ∀u⃗^D ∈ Ã then
12          do nothing to Ã
13        else
14          Ã = Ã − u⃗^D, for x⃗_n^D ≺ u⃗^D
15          Ã = Ã ∪ x⃗_n^D
16        end if
17        if (x⃗_n^D ⪯ p⃗_n^D) or (x⃗_n^D ≺≻ p⃗_n^D)
18          p⃗_n^D = x⃗_n^D
19        end if
20      end for
21    end for
```

After evaluating the objective vector for each particle in Line 10, we update the non-dominated archive and its individual best solution. If the particle is not weakly dominated by all the non-dominated solutions in the archive, then it is added into the archive and the solutions in the archive that are strongly dominated by the particle are removed from the archive (Lines 11–16). Finally, if the particle weakly dominates its individual best solution or they are not strongly dominated by each other, then the individual best solution is set to be the current position (Lines 17–19). The algorithm stops when the number of iterations has reached a pre-specified maximum.

4.2. Ranking process

Having used MOPSO to synthesize a range of solutions, a single solution must be selected by the decision maker. Ranking methods can be used to reduce the non-dominated solution set to a single design for (r, Q) inventory system. In this paper, a ranking process called TOPSIS is used for solving this kind of multi-attribute decision making (MADM) problem. TOPSIS is based on the concept that the best alternative should have the shortest distance from the positive-ideal solution and the longest distance from the negative-ideal solution (Hwang & Yoon, 1981). The reasons of using TOPSIS are the concept behind TOPSIS is rational and comprehensible, and the computation involved is simple (Deng, Yeh, & Willis, 2000).

A MADM problem with p alternatives each having q attributes can be mapped into a q-dimensional space with p points. We treat the non-dominated solutions as alternatives and use the three objectives, $[C(Q, k), N(Q, k), S(Q,k)]$, as attributes. The following data matrix, $\widetilde{R}$, generated from MOPSO is used to perform the computation of TOPSIS.

$$\widetilde{R} = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_p \end{array} \begin{array}{c} \begin{array}{ccc} C & N & S \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots \\ x_{p1} & x_{p2} & x_{p3} \end{bmatrix} \end{array}, \tag{11}$$

where x_{ij} is the jth attribute value (or objective function value) of alternative (or non-dominated solution) i. For the detailed steps of TOPSIS, please refer to Yoon and Hwang (1995).

5. Experimental results

The pharmaceuticals data shown in Table 2 is used to demonstrate the multi-criteria decision framework for inventory planning problems. All computation is executed on a laptop computer with Pentium M-1.6 GHz CPU and 1024 MB RAM. The first job is to set up various parameters for specific problem to get a better performance. These parameters include the number of iterations, number of particles, and maximum velocity of each variable (it is denoted as a percentage of the upper bound of each decision variable).

Table 2
The pharmaceutical data

ID	D	σ_L	A	c	h
1	3412	53.354	80	27.5	0.26
2	490	5.027	80	241	0.3
3	4736	57.911	135	29.41	0.3
4	200	2.969	80	233	0.26
5	215	2.781	80	435	0.3
6	22774	245.333	135	12.6	0.26
7	10557	85.395	135	2.14	0.26

5.1. Parameter tuning test

Stability and efficiency are two major criteria of the performance for any algorithm. So, the parameter tuning task is to find the level of each parameter under which the results obtained by MOPSO show better performance. For example, the pharmaceutical data mentioned earlier is solved ten times under different number of iterations to calculate the coefficient of variation (CV) in TOPSIS-ranked first solutions and the average execution time in Table 3. These two measures are used to evaluate the stability and efficiency of MOPSO, respectively. Tables 4 and 5 show the corresponding results of the other two parameter tuning tests.

From Table 3, it is obvious that the more iteration is, the longer execution time is. But different numbers of iterations do not have a significant influence on the stability of the MOPSO. The coefficients of variation are relative

Table 3
Computational results under different number of iterations

No. of iteration	Coefficient of variation		Average execution time(s)
	CV_Q	CV_k	
50	2.51	5.08	7.8
75	1.72	5.52	12.5
100*	1.28	3.91	17.1
125	1.31	3.70	23.1
150	4.66	5.43	27.6

Table 4
Computational results under different number of particles

No. of particle	Coefficient of variation		Average execution time(s)
	CV_Q	CV_k	
40	28.93	10.85	11.2
60	10.59	6.75	12.0
80	4.92	6.40	13.1
100	4.80	5.94	13.7
120	4.32	3.91	15.1
140	4.68	4.64	15.9
160*	0.81	4.21	18.0
180	1.42	4.45	18.5
200	2.27	3.58	19.3

Table 5
Computational results under different maximum velocity

Maximum velocity	Coefficient of variation		Average execution time(s)
	CV_Q	CV_k	
50	10.38	12.86	17.1
100	1.08	4.33	17.1
200	2.91	3.23	17.0
300*	1.38	2.55	17.3

stable ($CV_Q < 1.31$ and $CV_k < 3.91$) when the number of iterations is around 100 or 125. Table 4 shows that the average execution time is positively related to the number of particles. However, the coefficient of variation of order size and safety factor did not decrease more until the number of particles increases to 160. Especially if the number of particles is under 60, the coefficients of variation are very large. For the maximum velocity of each particle (Table 5), three hundredth of each decision variable is chosen as a balance between the stable solutions and execution time. Therefore, the following parameters (indicated in the row with an asterisk of Tables 3–5) are used for solving the entire pharmaceutical data.

1. The number of iterations is set to be 100.
2. The number of particles in a swarm is set to be 160.
3. The maximum velocity is set to three hundredth of the upper bound of each decision variable.

5.2. Computational results

The computational results of the whole dataset are shown in Table 6. All the coefficients of variation are between 0 and 6.2. This implies that the algorithm is quiet stable for the pharmaceutical data. For the computational time, it is also acceptable that the average execution time is around 16–17 s. Fig. 2 is the interface of the multi-criteria decision system for inventory planning problems. Graphical representation of non-dominated solutions and tabular list of TOPSIS results facilitate the usefulness of the framework presented in this paper.

Table 6
Computational results of the pharmaceutical data

ID	Coefficient of variation		Average execution time
	CV_Q	CV_k	
1	1.42	5.02	17.7
2	2.78	3.69	15.5
3	0.98	3.32	17.0
4	0.00	2.10	16.6
5	5.56	6.20	15.9
6	1.05	3.32	16.1
7	0.60	1.89	16.0

Table 7 lists ten efficient solutions from 612 found by MOPSO. Except the order size and safety factor, each row also shows the values of three objectives and the TOPSIS score. Assuming an equal weight for each objective, the solution having the highest TOPSIS score indicated with an asterisk is the compromise solution which fits the decision maker's preference most. Due to the space limit, the non-dominated solutions can not be listed here completely. However, as shown in Table 7, these solutions provide helpful information for decision makers to adjust their control policy to react to the ever-changing condition.

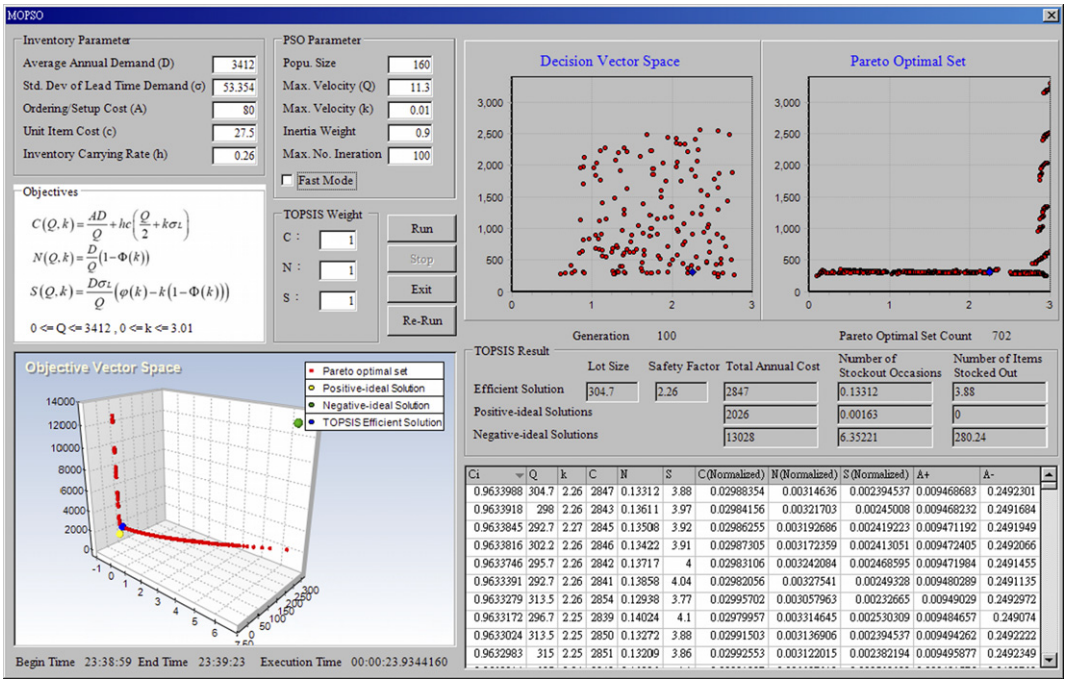

Fig. 2. The interface of the multi-criteria decision system for inventory planning problems.

Table 7
Non-dominated solutions of the ID 1 pharmaceutical data

TOPSIS Score	Q	k	$C(Q,k)$	$N(Q,k)$	$S(Q,k)$
0.7833	1929	2.76	8091	0.006	0.09
0.7953	1820	2.71	7690	0.007	0.12
0.8897	970	2.93	4867	0.007	0.04
0.9126	798	2.76	4248	0.013	0.21
0.9304	668	2.57	3777	0.027	0.63
0.9107	346	1.55	2617	0.596	18.58
0.9629*	299	2.38	2890	0.099	2.75
0.9135	262	1.7	2627	0.578	17.86
0.9599	262	2.74	3024	0.043	0.72
0.9568	235	2.9	3108	0.030	0.24

6. Conclusion

Traditional single-objective inventory planning models require the knowledge of the stock-out cost or service level. The difficulty in estimation of these parameters has been generally acknowledged. Hence the stage is set for a multi-objective analysis of the problem. Unfortunately, most multi-objective inventory models have been solved by aggregation methods through a linear combination of specific weights or only one objective was optimized and the others were turned into constraints. The reason to make such transformations is its merit in computational attractiveness. But the solutions obtained therein are unsatisfactory because managers make their decisions through a surrogate variable with incomplete information. A two-stage multi-criteria decision which consists of MOPSO and TOPSIS has been implemented to address the inventory planning problem. In the first stage, decision makers set up the parameters needed in MOPSO to simultaneously determine the non-dominated solutions of order size and safety factor without using any surrogate measure (e.g. service level or shortage cost) in a multi-objective context. Next, according to the weights reflecting the preference on each objective, TOPSIS is used to rank the non-dominated solutions generated in the first stage. Decision makers could tune up their preference to find the compromise solution that they are satisfied most. Numerical examples show that MOPSO performs well in solving MOOP through the information sharing mechanism among the particles.

To our best knowledge, this is the first time that the PSO is applied to the multi-objective inventory planning problems. Fields in multi-objective combinatorial optimization, such as job shop scheduling and vehicle routing, are suitable for this effective meta-heuristic. However, algorithm accommodating more complex constraints, such as linear or nonlinear functions, need to be invented before applying MOPSO to those optimization problems mentioned above. Other extensions are that the non-dominated set under consideration should be adequately small and meaningfully represent the complete Pareto front. Ideally, the set should emphasize the regions that entail significant tradeoff, and deemphasize the regions of little tradeoff.

References

Agrell, P. J. (1995). A multicriteria framework for inventory control. *International Journal of Production Economics, 41*, 59–70.

Alvarez-Benitez, J. E., Everson, R. M., & Fieldsend, J. E. (2005). A MOPSO algorithm based exclusively on Pareto dominance concepts. *Lecture Notes in Computer Science, 3410*, 459–473.

Coello-Coello, C. A., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. *Proceedings of the IEEE Congress on Computational Intelligence*, pp. 12–17.

Coello-Coello, C. A., & Pulido, G. T. (2001). Multiobjective optimization using a micro-genetic algorithm. *Proceedings of Genetic and Evolutionary Computation Conference* (GECCO'2001) (pp. 274-282). San Francisco, CA.

Coello-Coello, C. A., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 256–279.

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. Chichester: John Wiley and Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*, 182–197.

Deng, H., Yeh, C.-H., & Willis, R. J. (2000). Inter-company comparison using modified TOPSIS with objective weights. *Computers and Operations Research, 27*, 963–973.

Eberhart, R. C., & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. *Lecture Notes in Computer Sciences, 1447*, 611–616.

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *Proceedings of IEEE Congress on Evolutionary Computation* (pp. 81–86). Piscataway, NJ, Seoul, Korea.

Hwang, C.-L., & Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. Berlin/Heidelberg/New York: Springer-Verlag.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks* (Vol. IV, pp. 1942–1948). Piscataway, NJ, Seoul, Korea.

Knowles, J. D., & Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation, 8*, 149–172.

Liu, G. P., Yang, J. B., & Whidborne, J. F. (2003). Multiobjective Optimisation and Control. Hertfordshire: Research Studies Press, Chap. 4.

Mahapatra, N. K., & Maiti, M. (2005). Multi-objective inventory models of multi-items with quality and stock-dependent demand and stochastic deterioration. *AMO-Advanced Modeling and Optimization, 7*(1), 69–84.

Mandal, N. K., Roy, T. K., & Maiti, M. (2005). Multi-objective fuzzy inventory model with three constraints: A geometric programming approach. *Fuzzy Sets and Systems, 150*, 87–106.

Padmanabhan, G., & Vrat, P. (1990). Analysis of multi-item inventory systems under resource constraints: A non-linear goal programming approach. *Engineering Cost and Production Economics, 20*, 121–127.

Roy, T. K., & Maiti, M. (1998). Multi-objective inventory models of deteriorating items with some constraints in a fuzzy environment. *Computers and Operations Research, 25*(12), 1085–1095.

Shi, Y., & Eberhart, R. C. (1998a). *Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proceedings of EP 98*. New York: Springer (pp. 591–600).

Shi, Y., & Eberhart, R. C. (1998b). *A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press (pp.69–73).

Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling*. New York: John Wiley and Sons.

Yoon, K. P., & Hwang, C.-L. (1995). *Multiple Attribute Decision Making: An Introduction*. SAGE.