MULTIOBJECTIVE PSO ALGORITHM IN PROJECT STAFF MANAGEMENT

I-Tung Yang
Dept. of Civil Engineering
Tamkang University
151 Yingchuan Road, Tamsui, Taipei County, Taiwan 251
ityang@mail.tku.edu.tw

Abstract: The assignment of staff to projects is a regular activity for many contractors, government agencies and consulting firms. Whereas the primary objective of the assignment is to maximize the overall profit, issues dealing with manpower management must also be incorporated to ensure strong morale and to enhance competitiveness. Such issues include to avoid excessive overtime and to balance workloads. The present study develops a particle swarm optimization (PSO) algorithm to handle the assignment problem with multiple objectives, which creates difficulty for conventional optimization techniques. The proposed algorithm is tested on an application case to illustrate its performance. It has also been compared to LINGO, a commercial optimization package.
Keywords: PSO, Multiobjective optimization, Assignment problem, Manpower management

1. INTRODUCTION

One of the most important management tasks facing the construction industry is to assign the working staff to incoming projects. This routine task serves as the basis of profitability and, if done properly, can greatly enhance competitiveness. Yet, the staff-to-project assignment is usually done under time pressure.

To address manpower management issues, the real-world staff-to-project assignment involves multiple goals, such as (1) maximizing the overall profit, (2) minimizing the excessive overtime, and (3) minimizing the variation of workloads. The first goal is self explanatory. The second serves to balance the workloads of employees while the third intends to avoid excessive overtime hours. The latter two are considered important because they ensure morale and address non-financial motivation [5].

The assignment problem with multiple objectives has been proved to be NP-complete and even #P-complete [4]. That is, neither does polynomial-time algorithm exist nor is any deterministic algorithm kown to find the approximate answer within a reasonable error bound. To solve such a difficult problem in limited time, this study develops a particle swarm optimization (PSO) algorithm, supported by stochastic and iterative search, to optimize all the objectives simultaneously.

2. BACKGROUND

2.1 Mono-objective Assignment Problem

The original assignment problem is to assign n people to n jobs so as to reach some overall level of competence, i.e., to minimize or maximize an objective. The implied assumption is that each person would be assigned a job and each job would utilize exactly one person. For example, if the cost of assigning person i to job j is a_{ij}, the objective is to minimize

$$\sum_{i,j} a_{ij} x_{ij} \tag{1}$$

The binary constraints are

$$\sum_{i} x_{ij} = 1 \ \ \forall j \tag{2}$$

$$\sum_{j} x_{ij} = 1 \ \ \forall i \tag{3}$$

where the assignment variable x_{ij} is 1 if person i is assigned to job j and 0 otherwise.

2.2 Solution Procedure

After its initial appearance in 1950s, the mono-objective assignment problem has been tackled by several approaches. On one hand, the problem can apparently be regarded as binary integer programming because x_{ij} can only take the value 0 or 1. On the other hand, it has been shown that by relaxing the integer assumption, linear programming techniques can also lead to the desired result [8]. Although both integer and linear programming models can be solved by commercial software packages, the models could be tediously large since it demands n^2 variables.

It is more convenient to use a specialized algorithm. One of the famous approaches is the Hungarian algorithm [7]. Also of help is the minimum cost flow algorithm [1], which sends flow from a set of supply nodes through the arcs of a network, to a set of demand nodes, at minimum total cost. Each arc in the network denotes an assignment between staff and projects.

2.1 Multi-objective Assignment Problem

Despite the success in achieving the single goal for the assignment problem, recent attentions have been shifted to optimizing multiple objectives simultaneously. Whereas the primary objective is to minimize costs or maximize profits, managers are often concerned with unbalanced workloads and excessive overtime. The former raises conflicts between heavily and lightly loaded teams [9] and the latter creates

stress and fatigue, and ultimately causes poor-quality products.

To deal with multiple objectives, several methods are at hand. The most intuitive one is taking the weighted sum of all the objectives and optimizing the weighted sum instead. The weighted sum approach, however, is not appropriate here because the objectives of the assignment problem are measured in different units and their relative weights are often difficult to assess.

Another approach is goal programming, which intends to minimize the weighted sum of the absolute deviations between pre-specified goals and objective values. Again, this approach requires decision makers to devise goals for objective values and set a proper weight for every objective before the optimization. The tasks rely on a priori articulation of preference information, which may not be accessible to decision makers.

The third approach is to optimize the most relevant objective, and considering other objectives as constraints bound by some allowable levels, ε. The major shortcoming of the ε-constraint approach is that it has to tediously repeat the optimization procedure for different bounds of ε, if a full description of optimal solutions is needed.

The optimality of solutions in multiobjective optimization is based on a tradeoff philosophy. That is, we are really trying to find good compromises (the "trade-off surface" in the search space) rather than a single solution. The tradeoff surface is composed of non-dominated, also known as Pareto optimal, solutions that are better than other solutions at some objectives while being at least as good as others for the other objectives. Having the tradeoff surface, as opposed to a single solution, offers decision makers the most flexibility for determining the compromised assignment alternative after a thorough evaluation of existing solutions. Thus, the ultimate aim of the present study is to approximate the tradeoff surface for the multiobjective assignment problem. Detailed definitions of multi-objective dominance can be found in [3].

3. PROBLEM STATEMENT

The targeted problem is to assign n staff teams to m incoming projects over a planning time horizon. Note that the numbers of teams and projects need not be equal. Hence, in addition to assigning teams to projects, the solution would also help determine which project should be accepted.

Five sets of entries are needed. The main entry is the estimated man-hours for team i to perform project j, denoted by t_{ij}. For the same project, the required man-hours may vary. This is to accommodate an allowance for added productivity when a team is familiar with a particular project. Thus the optimal assignment plan will encourage teams to perform their specialty, using less time. The vector r_j denotes the revenue of project j. Three vectors, c_i, s_i, and g_i represent the hourly cost, the availability (upper limit of working hours), and the regular work time for team i, respectively. The availability differs because some of the teams are concurrently working on other in-house projects, which shall not be reassigned to avoid disruption. Working hours beyond the regular work time are overtime and would incur a higher cost, e.g., 1.5 times the regular hourly cost. By definition, the regular work time is not greater than the availability.

As mentioned previously, the decision variables are the binary assignment choices between team i and project j

$$x_{ij} = 0 \text{ or } 1 \tag{4}$$

The present model consists of three objectives, whose individual reasoning and formulation are given below.

The first objective is to maximize the profit, which is revenue R minus the regular and overtime costs, C_{regu} and C_{over}.

$$P = R - C_{regu} - C_{over} \tag{5}$$

The revenue can be expressed as

$$R = \sum_{\forall j} r_j \left(\sum_{\forall i} t_{ij} x_{ij}\right) \tag{6}$$

where $\sum_{\forall i} t_{ij} x_{ij}$ stands for the actual working hours spent on project j; r_j is the revenue of project j. The regular-time cost is

$$C_{regu} = \sum_{\forall i} c_i [\min(\sum_{\forall j} t_{ij} x_{ij}, g_i)] \tag{7}$$

where $\sum_{\forall j} t_{ij} x_{ij}$ denotes the actual working hours of team i, which is the minimum between the actual working hours and the regular time g_i. It is then multiplied by the regular hourly cost c_i to obtain the regular-time cost. The overtime cost is

$$C_{over} = \sum_{\forall i} c_i^* [\max(0, \sum_{\forall j} t_{ij} x_{ij} - g_i)] \tag{8}$$

where c_i^* is the hourly rate of overtime pay. The element in square brackets calculates the overtime hours by deducting the regular work time from the actual working hours, if the latter is greater.

The second objective is to minimize the variation of workloads. For each team, the workload is quantified by a standardized utilization rate: dividing the team's actual working hours by the regular work time:

$$u_i = \frac{(\sum_{\forall j} t_{ij} x_{ij})}{g_i} \tag{9}$$

The utilization rate is less than 100% when the assigned projects can be completed within the regular time; it is greater than 100% at the presence of overtime. On the foundation of Eq. (9), the variation of workloads is defined to be the standard deviation of utilization rates:

$$\sigma = \sqrt{\frac{1}{n} \sum_{\forall i} (u_i - \bar{u})^2} \tag{10}$$

where $\bar{u}$ is the average utilization rate

$$\bar{u} = \frac{1}{n} \sum_{\forall i} u_i \tag{11}$$

The third objective is to minimize excessive overtime, which is the maximal amount among all teams' overtime hours.

$$h = \max_{\forall i}[\max(0, \sum_{\forall j} t_{ij} x_{ij} - g_i)] \quad (12)$$

4. PSO ALGORITHM

4.1 Concepts

The original PSO scheme was designed to mimic the cooperation within a biological population, such as a group of birds or a swarm of insects [6]. Within the population, multi-dimensional particles, each a possible solution, are flown through the problem space, in search of optima. Each particle has its own velocity, which is determined by (1) the local best: the memory of the best solution it has obtained thus far and (2) the global best: the best solution found by the entire population. It has been shown that PSO is able to converge to global optima fast without being trapped in local optima, especially when the problem space is complex and irregular.

The capability of PSO algorithms has been testified by a wide variety of recent applications, such as biomedical image registration [11], neural network training [2], and resource-constrained scheduling [14]. However, the original PSO scheme focuses on only one objective and hence requires further enhancements in the context of multiobjective optimization. In what follows, we introduce the improved PSO algorithm

4.2 Proposed Algorithm

In the proposed PSO algorithm, the position of the *k*th particle is *m*-dimensional (*m* equals the number of projects) and expressed in a vector form: $\mathbf{Y}^k = [y_1^k, y_2^k, ..., y_d^k ..., y_m^k]$, where all $y_d^k \in [0,1]$. The component in dimension *d* of the position determines which team is assigned to project *d* by a mapping process. The process is virtually mapping a continuous variable y_d^k to a binary variable x_{ij} as graphically depicted in Fig. 1 where *m* equals 3.

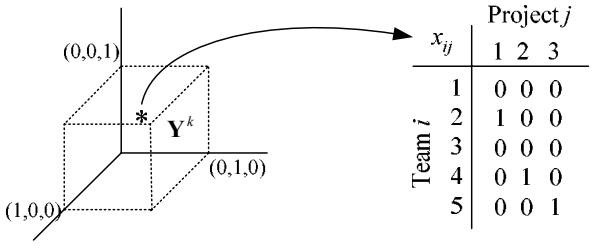

Fig. 1. Mapping process

The proposed algorithm iteratively changes the particle positions as follows

$$y_d^k(t+1) = y_d^k(t) + v_d^k(t+1) \quad (13)$$

where $y_d^k(t)$ is the component in dimension *d* of particle *k* at iteration *t*; $v_d^k(t+1)$ is the velocity at iteration *t*+1, which is determined by

$$v_d^k(t+1) = wv_d^k(t) + c_1 r_1 (Lbest_d - y_d^k(t)) + c_2 r_2 (Gbest_d - y_d^k(t)) \quad (14)$$

where w is the inertia weight; $v_d^k(t)$ is the velocity in the previous iteration; c_1 and c_2 are learning constants; r_1 and r_2 are random factors in the [0,1] interval; $Lbest_d$ is the component in dimension *d* of the local best; $Gbest_d$ is the component in dimension *d* of the global best. While the swarm size, w, c_1, and c_2 are specifiable algorithm parameters, r_1 and r_2 provide the imperative randomness for finding better solutions along the direction guided toward the local and global best.

The algorithm parameters in Eq. (14) should be fine-tuned to ensure performance. However, previous experiments have suggested the following configurations. The swarm usually contains 10 to 50 particles. The inertia weight w is used to control the impact of the previous history of velocities on the current velocity, thus to influence the trade-off between global exploration and local exploitation abilities of the particles. It is therefore better to initially start at a large value (around 1), in order to promote global exploration, and gradually decrease it (no less than 0) to get more refined solutions [10]. Both learning constants c_1 and c_2 range from 1 to 4, whereas relative importance can be given to stress the influence of one's own memory (cognition learning) or that of the entire population (social learning).

The velocity is constrained within a specified bound to avoid vicious oscillation

$$v_d^k(t+1) = \frac{v_d^k(t+1)}{|v_d^k(t+1)|} v^{\max} \quad \text{if } |v_d^k(t+1)| > v^{\max} \quad (15)$$

where the velocity bound $v^{\max}$ is often smaller than the domain of the search space.

Besides the velocity bound, the particle positions are constrained within the feasible range [0,1]. This is accomplished by a new bouncing routine illustrated in Fig. 2. Once a particle is moved beyond the feasible range, it will be automatically sent back with a distance equal to that beyond the limit. The bouncing routine is considered superior to the conventional absorbing routine, i.e., the particle is kept at the boundary as its velocity is absorbed (shown as the gray circle), because it provides more exploration capability.

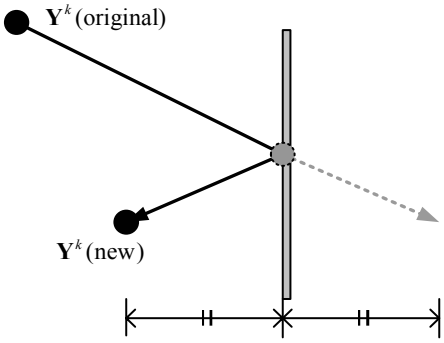

Fig. 2. Bouncing routine

A key requirement for locating the non-dominated solutions is to preserve solution diversity during optimization. To do so, we maintain an elite archive and alter the definition of the global best in the original PSO scheme. As iterations progress, the global best is selected dynamically from the elite archive that stores all the non-dominated solutions. The rule of selection gives preference to the non-dominated solutions that dominates the "fewest" particles in the current iteration. The underlying concept is to preserve diversity by promoting movements to the extremes and unrepresented areas.

The local best is the best position ever achieved by the particle. Every particle compares its current position to the previous local best and chooses the non-dominated one as the new local best.

The proposed PSO algorithm is not sensible to the scales of the objectives because it by no means calculates the distances between solutions. Thus its performance is independent from the choice of scales. For instance, converting the profit from local currencies to US dollars does not affect the solutions being found. This is of particular importance to international firms.

The proposed PSO algorithm is composed of the following steps

1. Randomly initialize the positions for all the particles.
2. Initialize the velocity of each particle.
3. Map the particle positions to assignment choices.
4. Evaluate the four objective values of the assignment choices according to Eqs. (5), (10), and (12).
5. Store the positions representing non-dominated solutions in the elite archive.
6. Initialize the memory of each particle.
7. WHILE the maximal number of iterations has not yet been reached
   For each particle DO
   (a) Compute the velocity as described in Eq. (14).
   (b) Constrain the velocity so that it does not exceed the bound by Eq. (15).
   (c) If the velocity would cause infeasibility, adjust it using the bouncing routine.
   (d) Compute the new position by adding the velocity to the previous position using Eq. (13).
   (e) Evaluate the objective values of the current position.
   (f) Compare the new position with members in the archive.
   (g) Update the elite archive by inserting all the currently non-dominated positions and eliminate any dominated locations from the archive.
   (h) When the new position dominates the local best, replace the local best.
   (i) Locate the archive member that dominates the fewest particles in this iteration as the global best.
   (j) Increase the loop counter.
8. Return the archive as the non-dominated solution set.

5. APPLICATION

An actual case was used to demonstrate the application of the proposed algorithm. This case is adopted from a previous study of an environmental consulting engineering firm [13]. Six engineering teams are available to be assigned to fifteen projects. Table 2 (at the end of this article) enumerates input data in the following sets:

(1) t_{ij}: estimated man-hours for team i to perform project j;
(2) r_j: revenue for project j;
(4) c_i: hourly cost of team i;
(5) s_i: availability of team i;
(6) g_i: regular work time for team i.

The overtime hourly cost is 1.5 times of the regular hourly cost.

There are three objectives being optimized: (1) to maximize the overall profit, (2) to minimize the maximal overtime, and (3) to minimize the variation of workloads. Detailed formulas of these objectives can be found in Section 3.

The multiobjective assignment problem is solved using the following algorithm parameters:

(1) Swarm size is 20;
(2) Inertia weight starts at 1.0 and linearly decreases to 0.4;
(3) Learning constants are 1 and 2 (social learning is twice more important than cognition learning);
(4) Velocity bound is 0.5.

A pilot study is conducted to set the iteration number. In summary, the maximal iteration number is set to 10,000 as it yields the most non-dominated solutions per CPU time. This criterion is one of the performance metrics in multiobjective optimization [15].

A set of non-dominated solutions are found within 15 minutes at a Pentium IV 2.4 GHz machine. Fig. 3 plots the non-dominated solutions, which reflect a three-dimensional

tradeoff between profit, overtime, and variation of workloads.

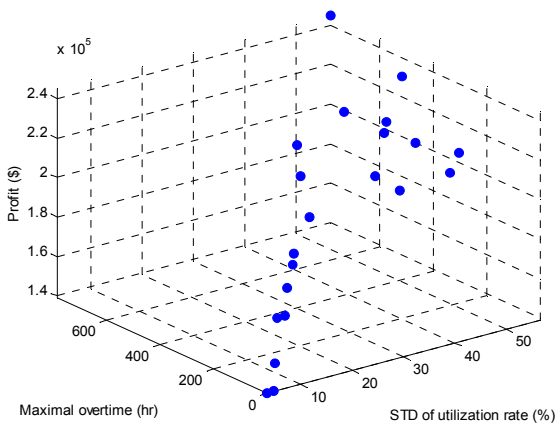

Fig. 3. Non-dominated solutions

Due to limited space, Table 3 lists only the first 10 solutions, sorted by profit. A close observation confirms that a higher profit comes at the expense of more excessive overtime or a greater variation of workloads.

Table 3. Non-dominated solutions (partial list)

Profit ($)	Maximal overtime (hr)	SD of utilization rate (%)
$244,915	785	56.71
$243,616	**383**	**49.79**
$241,973	185	36.39
$239,411	325	35.47
$236,633	190	36.15
$234,973	0	40.90
$234,841	263	23.04
$232,518	138	39.46
$229,422	50	27.25
$225,850	0	39.10

The non-dominated solution set illustrates the advantage of the proposed algorithm: giving decision makers a set of compromised solutions to choose from when time is limited. For instance, whereas the second highest profit (marked in bold) is about $1300 less, it can decrease the maximal overtime from 785 to 383 hours and also lead to a more balanced workload. It is therefore chosen to be the optimal solution after all.

To validate the effectiveness of the proposed algorithm, it is compared to LINGO 8.0, a powerful optimization package [12]. Since LINGO cannot handle multiple objectives, we used it to maximize the profit under the following two constraints: (1) the maximal overtime is less than 383 hours, and (2) the standard deviation of utilization rates is less than 49.79%. LINGO took 6 million iterations, using the branch-and-bound solver, to find the local optimal solution of $241,751, which is apparently dominated by our solution: $243,616.

In addition to the effectives, the proposed algorithm is notable for its efficiency. For the application case, a full permutation would require $O(10^{12})$ trails. The huge number of trials is obviously expensive and almost infeasible in computation. In contrast, the proposed algorithm tries only $O(10^5)$ possible solutions (20 particles for 10,000 iterations). In a nutshell, the proposed algorithm searches a very small portion of the solution space (in the order of 10^{-7}) and yet, finds very competitive non-dominated solutions.

6. CLOSING REMARKS

The present study develops a new PSO algorithm to facilitate the multi-objective staff-to-project assignment, which cannot be solved in polynomial time using conventional optimization techniques. Thus, the aim of the proposed algorithm is to locate good non-dominated solutions under time pressure. To optimize multiple objectives simultaneously, the proposed algorithm maintains an elite archive and uses the archive members to dynamically lead the particle swarm in searching for more and better non-dominated solutions. The performance of the proposed algorithm has been validated through an application case. It has been shown that the proposed algorithm can find very competitive solutions with considerable efficiency.

REFERENCES

[1] Ahuja, B.K., Magnanti, T.L., & Orlin, J.B. (1993). *Network flows: theory, algorithms, and applications*, Prentice-Hall, Englewood Cliffs, NJ, 470-472.

[2] Chatterjee, A., Pulasinghe, K., Watanabe, K., & Izumi, K. (2005). "A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems**."** *IEEE Transactions on Industrial Electronics*, 52(6), 1478-1489.

[3] Coello Coello, C.A., Veldhuizen, D.A., & Lamont, G.B. (2002). *Evolutionary algorithms for solving multi-objective problems*, Kluwer Academic Publishers, NY.

[4] Ehrgott, M. (2005). *Multicriteria optimization*, 2nd edition, Springer, New York, 253-267.

[5] Hecker, P. (1996). "Human resources strategies for successful consulting engineering firms." *Journal of Management in Engineering*, 12(5), 32-36.

[6] Kennedy J., Eberhart R.C., & Shi Y. (2001). Swarm intelligence, Morgan Kaufmann Publishers, San Francisco, CA.

[7] Kuhn, H.W. (1955). "The Hungarian method for the assignment problem." *Naval research Logistics Quarterly*, 2, 83-97.

[8] Luenberger, D.G. (1984). *Linear and nonlinear programming*, 2nd edition, Addison-Wesley, Reading, MA, 133-134.

[9] Molleman, E. (2005). "The multilevel nature of team-based work research." *Team Performance Management*, 11(3), 113-124.

[10] Shi, Y. & Eberhart, R. (1998). "Parameter selection in particle swarm optimization." *Proceedings of the Seventh Annual Conf. on Evolutionary Programming*, 591-601.

[11] Wachowiak, M. P., Smolikova, R., Zheng, Y., Zurada, J. M., & Elmaghraby, A. S. (2004). "An approach to multimodal biomedical image registration utilizing particle swarm optimization." *IEEE Transactions on Evolutionary Computation*, 8(3), pp. 289-301.

[12] Winston, W.L. & Venkataramanan, M. (2002). *Introduction to mathematical programming - applications and algorithms*, 4th edition, Duxbury Press, Boston, MA.

[13] Zanakis, S.H. (1983). "A staff to job assignment (partitioning) problem with multiple objectives." *Computer & Operations Research*, 10(4), 357-363.

[14] Zhang, H., Li, X., Li, H., & Huang, F. (2005). "Particle swarm optimization-based schemes for resource-constrained project scheduling." *Automation in Construction*, 14(3), pp. 393-404.

[15] Zitzler, E., Deb, H., & Thiele, L. (2000). "Comparison of multi-objective evolutionary algorithms: Empirical results." *Evolutionary Computation*, 8(2), 173-195.

Table 2. Input data

	t_{ij}	Project															c_i ($)	s_i (hr)	g_i (hr)
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
Team	1	810	877	930	545	430	515	423	720	410	515	410	480	250	345	345	56.4	2175	1450
	2	810	965	930	600	390	515	465	720	410	468	410	480	275	345	345	38.0	2790	1860
	3	810	965	930	600	430	515	465	720	373	515	410	480	275	345	345	37.0	2790	1860
	4	810	965	930	600	430	515	465	720	410	515	410	480	275	345	345	34.0	2790	1860
	5	810	965	930	600	430	515	465	720	373	515	410	480	275	345	345	51.6	1245	830
	6	810	965	930	600	430	515	465	655	410	515	410	480	275	314	345	42.4	930	620
r_j ($1000)		61.2	65.2	35.2	36.8	34.2	35.8	39.2	34.8	35.4	42.6	34.2	25.0	23.0	36.8	33.8			