Multi-Objective Optimization for Software Development Projects

Tad Gonsalves and Kiyoshi Itoh, *Members, IEEE*

***Abstract*— The development of software projects requires the co-ordination of the efforts of a team of professionals with varying skills. The major problem in software development is to assign the right personnel to the right job at the right time and at the right cost. In this study, we propose a skill-to-time model in which the task processing time varies in accordance with the level of skill possessed by the personnel assigned to the task. We cast the problem as a multi-objective optimization problem and simultaneously optimize the development cost and the development time using the Multi-Objective Particle Swarm Optimization algorithm. The optimization result is a well-defined and well-spread Pareto front representing the trade-off between development time and development cost.**

***Index Terms*— Multi-Objective Optimization, Multi-Objective Particle Swarm Optimization**

I. Introduction

A. Skill-based allocation

The problem faced by the management in software development is one of human resource allocation, or staffing, i.e., given a group of available developers and a set of project activities, which developers to activity allocation yields more value to the organization? In this paper, we propose a skill-based allocation scheme in which personnel are allocated to the various tasks in a software development project in accordance with their skills. The study illustrates the use of the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm in simultaneously minimizing the development cost and the development time.

The skill-based allocation of personnel to the tasks in software development projects is a variation of the well-known scheduling problems. The scheduling problems focus their attention on the time-table, shifts, breaks, holidays – with the implicit assumption that a given pool of personnel possess identical level of skills. The skill-based allocation problem we consider in this study focuses its attention on the allocation of staff to the tasks taking into consideration the matching of the staff skills to those required for the processing of tasks. Several research studies have been devoted to the staffing problem. Baretto et al. address project staffing as a constraint satisfaction problem [10] and define optimizable utility functions, while Acuna and Juristo deal with the effect of team performance on the overall development result [1].

It is extremely difficult to find good solutions to these highly constrained and complex problems and even more difficult to determine *optimal* solutions that minimize costs, meet employee preferences and satisfy all the workplace constraints. Different solution techniques are in use – mathematical programming techniques [6], [11], [12], [28], heuristics [7], [8], [19], [22], [36] and the AI approach of using meta- heuristics [2], [3]. Finding an optimal solution to the skill-based staffing problem proposed in this study constitutes a multivariate, multimodal, combinatorial optimization problem involving many constraints, which clearly, cannot be solved by exhaustive methods.

A major goal of software engineering is to produce higher quality software, while keeping effort expenditure and schedule time to a minimum. This is the motivation of our research. We design the MOPSO algorithm in such a way that it allocates the personnel to the development tasks which minimizes the development cost and the development time. The Particle Swarm Optimization (of which MOPSO is a variation) is a simple yet powerful biologically inspired swarm intelligence paradigm [30], [31] that is rapidly gaining importance in Computational Intelligence [20], [21]. Its applications are also rapidly expanding [31]. To the best of our knowledge, this study is the first attempt to apply the MOPSO meta-heuristic to optimize the software development projects cost and time simultaneously.

B. Multi-Objective Optimization

Most real-world optimization problems have multiple objectives which are often conflicting. The goal of multi-objective optimization (MOP) is to optimize the conflicting objectives simultaneously. In this section, we define the general form of a MOP and Pareto dominance for identifying optimal solutions. Towards the end of the section, we describe the use of Evolutionary Algorithms to solve MOP problems.

Most multi-objective optimization algorithms use the concept of domination. In these algorithms, two solutions are compared on the basis of whether one solution dominates the other or not. Assume that there are M objective functions to be optimized and the problem is one of minimization. A solution $x^{(1)}$ is said to dominate the other solutions $x^{(2)}$, if conditions (1) and (2) are both true [17].

Manuscript received December 30, 2009. This work has been supported by the Open Research Center Project funds from "MEXT" of the Japanese Government (2007-20011).

T. Gonsalves is with the Department of Information & Communication Sciences, Sophia University, Tokyo, Japan (phone: 81-3-3238-4143; fax: 81-3-3238-3311; e-mail: t-gonsal@ sophia.ac.jp).

K. Itoh is with the Department of Information & Communication Sciences, Sophia University, Tokyo, Japan (e-mail: itohkiyo@ sophia.ac.jp).

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, or $f_j(x^{(1)}) \not> f_j(x^{(2)})$ for all $j = 1, 2,\ldots M$.

2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, or $f_j(x^{(1)}) < f_j(x^{(2)})$ for at least one j belonging to {1,2,..M}.

If either of the above conditions is violated, the solution $x^{(1)}$ does not dominate the solution $x^{(2)}$. The non-dominated solutions give rise to a Pareto front. The points on the front are used to select a particular combination of functions that are in a trade-off balance.

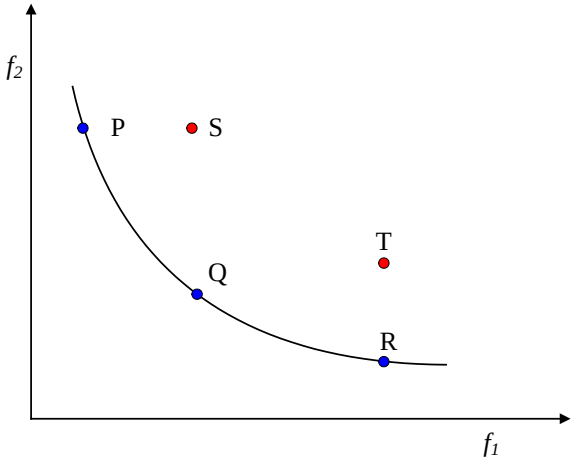

Fig.1. Pareto Dominance (min-min problem)

Figure 1 illustrates the concept of Pareto dominance for the minimization of two objective functions. Solution S is dominated by solution P in the f_1 objective, while solution T is dominated by solution R in the f_2 objective. The solutions P, Q, R are Pareto-optimal solutions since none of them is dominated by any other solutions.

II. SKILL-BASED STAFF ALLOCATION

In this section, we explain the skill-based allocation approach in which personnel are assigned to the development tasks in accordance with their skills. The assignment according to skills can result in the increase of the expected task processing time in case of ill-assignment.

A. *Skill levels of the staff*

Each personnel hired or employed by the management for the software development project possesses a varying amount of skills. The skills represent the knowledge, expertise and capabilities of the staff in database, Java programming, teamwork, object-oriented design, requirement elicitation techniques, test techniques, etc. The skill levels are within the 1 ~ 5 range as shown in Table 1. The cost per day of the personnel is also listed in Table 1.

Table 1 Developmental personnel and their skill levels

Developmental skills	Software development personnel and thier skill levels						
	P1	P2	P3	P4	P5	P6	P7
Database	3	3	4	3	3	3	4
Java	3		4	3	3	4	4
Negotiation	2	2	3		2		3
Team work	2	2	3	2	3		3
Object oriented design		3	4				4
Telecommunications		3	3				4
Object oriented analysis		3		4			4
Relationship with people		2	3				3
Testing			3	3			
Requirements elicitation				3			
Skill lelvels total (q)	10	18	27	18	11	7	29
Personnel cost/day (yen)	15,000	25,000	30,000	20,000	10,000	15,000	35,000

B. *Allocation according to skill level*

Each task in the development phase contains a list of skills that are required to perform that task. Moreover, each skill has a certain level that has to be met if the task is to be performed on time. Personnel are allocated to the tasks in accordance with their skill levels. In our skill-based allocation model, the task process time varies in proportion to the *match* between the skills required to perform the task and those possessed by the personnel. If q represents the skills required to perform the task and p the skills possessed, then the fractional difference between the skills required for the task and those actually possessed by the assigned personnel is given by:

$$\delta = (p-q)/q \quad ; \quad -1 < \delta \leq 0 \qquad (1)$$

The estimated processing time, is given by:

$$T = T_o \chi^{\delta} \qquad (2)$$

where, χ represents the *extension coefficient* determined by the domain experts' heuristics and T_o the original task processing time. Figure 2 shows the variation of T_o (=20 days) for three different values of χ. As shown in the diagram, the task process time decreases as the skills possessed by the staff allocated to the task match the skills required to process the task.

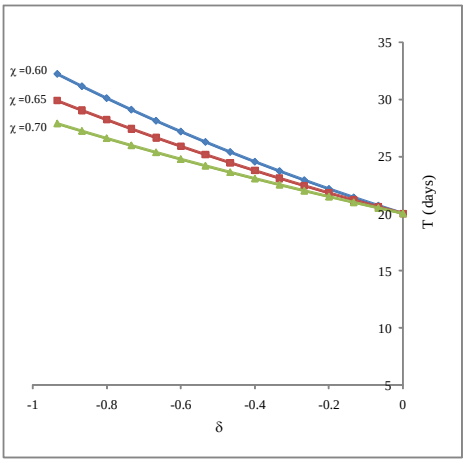

Fig.2 Skill-dependent expansion of the task processing time

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

III. Optimization problem formulation

In general, a multi-objective minimization problem with M decision variables and N objectives can be stated as:

$$\text{Minimize } f_i(\mathbf{x}) = 1,\ldots, N \quad \text{where } \mathbf{x} = (x_1, \ldots, x_m) \in X \tag{3}$$

$$\text{subject to : } \begin{cases} g_j(\mathbf{x}) = 0 & j = 1,\ldots, M \\ h_k(\mathbf{x}) \leq 0 & k = 1,\ldots, K \end{cases} \tag{4}$$

Here, f_i is the i^{th} objective function, $\mathbf{x}$ is the decision vector that represents a solution and X is the variable or parameter space. The functions g_j and h_k represent the equality and the inequality constraints, respectively. The desired solution is in the form of a "trade-off" or compromise among the parameters that would optimize the given objectives. The optimal trade-off solutions among the objectives constitute the Pareto front. MOP deals with generating the Pareto front, which is the set of non-dominated solutions for problems having more than one objective. A solution is said to be non-dominated if it is impossible to improve one component of the solution without worsening the value of at least one other component of the solution. The goal of multi-objective optimization is find the true and well-distributed Pareto front consisting of the non-dominated solutions.

A. Objective functions

Project development cost

If the cost per unit time of the j^{th} personnel is P_C and the total number of personnel allotted to the project development is m, then the total development cost is

$$f_1 = \sum_{j=1}^{m} P_{C\,j} \tag{5}$$

Project development duration

If P_{Tj} is the processing time of the j^{th} task in the project and Q_{Tj} is the queuing time of the j^{th} task in the project for resource availability, then the actual duration of the project is given by:

$$f_2 = \sum_{j=1}^{m} (Q_{Tj} + P_{Tj}) \tag{6}$$

B. Constraints

Task precedence relations

The series of tasks in the software development project have precedence relations among them. If S_{Tj} is the starting time of the j^{th} task and F_{Tj} the finishing time, then the precedence relation of the two consecutive tasks can be expressed as:

$$F_{T(j-1)} < S_{Tj} \quad \forall j \tag{7}$$

Personnel Availability

At a given time, personal can be allocated to a task only if they are available. The availability hard constraints imply that no personnel can be simultaneously allocated to more than one task at a time.

$$P_j(t) \Rightarrow \tau_i(t) \neq P_j(t) \Rightarrow \tau_k(t) \quad \forall i,j,k \tag{8}$$

III. Evolutionary algorithms & MOPSO

Evolutionary Algorithms (EA) seem to be especially suited to MOP problems, due to their abilities to search simultaneously for multiple Pareto optimal solutions and to perform better global searches of the search space. Many evolutionary algorithms have been developed for solving MOP. Examples are: GA [25], NSGA-II [16], a variant of NSGA (Non-dominated Sorting Genetic Algorithm) [39]; SPEA2 [43-44], which is an improved version of SPEA (Strength Pareto Evolutionary Algorithm); and PAES (Pareto Archived Evolution Strategy). These EAs are population-based algorithms that possess an in-built mechanism to explore the different parts of the Pareto front simultaneously. The Particle Swarm optimization (PSO), which was originally designed for solving single objective optimization problems, is also extended to solve multi-objective optimization problems. The extension of PSO to MOPSO (Multi-objective Particle Swarm Optimization) is found in [4], [13], [14], [23], [26], [27], [32-35], [37], [38].

A. PSO

The Particle Swarm Optimization (PSO) algorithm imitates the information sharing process of a flock of birds searching for food. The population-based PSO conducts a search using a population of individuals. The individual in the population is called the particle and the population is called the swarm. The performance of each particle is measured according to a predefined fitness function. Particles are assumed to "fly" over the search space in order to find promising regions of the landscape. In the minimization case, such regions possess lower functional values than other regions visited previously. Each particle is treated as a point in a d-dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of the other companion particles. By making adjustments to the flying based on the local best (*pbest*) and the global best (*gbest*) found so far, the
swarm as a whole converges to the optimum point, or at least to a near-optimal point, in the search space.

The notations used in PSO are as follows: The i^{th} particle of the swarm in iteration t is represented by the d-dimensional vector, $x_i(t) = (x_{i1}, x_{i2}, \ldots, x_{id})$. Each particle also has a position change known as velocity, which for the i^{th} particle in iteration t is $v_i(t) = (v_{i1}, v_{i2}, \ldots, v_{id})$. The best previous position (the position with the best fitness value) of the i^{th} particle is $p_i(t\text{-}1) = (p_{i1}, p_{i2}, \ldots, p_{id})$. The best particle in the swarm, i.e., the particle with the smallest function value found in all the previous iterations, is denoted by the index g. In a given iteration t, the velocity and position of each particle is updated using the following equations:

$$v_i(t) = wv_i(t\text{-}1) + c_1r_1(p_i(t\text{-}1) - x_i(t\text{-}1)) + c_2r_2(p_g(t\text{-}1) - x_i(t\text{-}1)) \tag{9}$$

and

$$x_i(t) = x_i(t\text{-}1) + v_i(t) \tag{10}$$

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

where, i =1, 2,…,NP; t = 1, 2,…,T. NP is the size of the swarm, and T is the iteration limit; c_1 and c_2 are positive constants (called "social factors"), and r_1 and r_2 are random numbers between 0 and 1; w is the inertia weight that controls the impact of the previous history.

B. MOPSO

The Multi-Objective Particle Swarm Optimization (MOPSO) is an extension of the single objective Particle Swarm Optimization (PSO). In PSO, gbest and pbest act as guides for the swarm of particles to continue the search as the algorithm proceeds. The main difficulty in MOPSO is to find the best way of selecting the guides for each particle in the swarm. This is because there are no clear concepts of pbest and gbest that can be identified when dealing with a set of multiple objective functions. Our algorithm is similar to the ones described in [4], [13], [14]. It maintains an external archive A, containing the non-dominated solutions found by the algorithm so far.

The algorithm begins with the initialization of an empty external archive, A (line 1 in Figure 4). The positions (xi) and velocities (vi) of a swarm of N particles are initialized randomly (line 2). The number of Perspectives assigned to each context and the service time of the contexts are randomly generated within the given bounds imposed by the management (Refer to Table 1 for the representation of a typical particle). The operation of the service system is simulated by means of the discrete event simulator and the values of f1 (service cost) and f2 (waiting cost) for each particle are computed using equations 5 and 6, respectively. The initial position of each particle is considered to be its personal best (Pi = xi) as well as the global best (Gi = xi).

At each iteration t the velocities of the positions of the particles are updated using equation 9. The velocities of each particle are forced into the feasible bounds, if they have crossed the lower or the upper bounds. Similarly, the particle positions are updated using equation 10 (lines 5-8). This is followed by the evaluation of the objective functions f1 and f2 for each of the particles (line 9). Any solutions which are not weakly dominated by any member of the archive are added to A (line 12) and any elements of A which are not dominated by xi are deleted from A.

The crucial parts of the MOPSO algorithm are selecting the personal and the global guides. If the current position of xi weakly dominates Pi or if xi and Pi are mutually non-dominating, then Pi is set to the current position (lines 15-17). Members of A are mutually non-dominating and no member of the archive is dominated by any xi. All the members of the archive are, therefore, candidates for the global guide. The algorithm pseudo-code is shown in Figure 3.

1: A := $\varnothing$
2: { x_i, v_i, G_i, P_i} i = 1,…,N
3: for t := 1 : G
4: for i := 1 : N
5: for k := 1 : K
6: $v_{ik} := wv_{ik} + r_1(P_{ik} - x_{ik}) + r_2(G_{ik} - x_{ik})$
7: $x_{ik} := x_{ik} + v_{ik}$
8: end
9: y_i := f (x_i)
10: if $x_i \not\preceq u \; \forall u \in A$
11: A := {u ∈ A | u $\not\prec$ x_i }
12: A := A U x_i
13: end
14: end
15: if $x_i \leq P_i$
16: $P_i := x_i$
17: end
18: G_i := select Guide(x_i , A)
19: End

Fig.3. MOPSO Algorithm

IV. OPTIMIZATION RESULTS USING MOPSO

The level of skills and time needed to process the tasks are shown in Table 2, while the optimized skill-based allocation of the personnel to the tasks, including the extension of the processing time as a function of the total level of skills, is shown in Table 3. The queuing time of each task is obtained by means of discrete event simulation [9], [24]. The total development time is the sum of the processing time and the waiting (queueing) time. The cost per personnel corresponds to the time the personnel spend on a task to which he/she is allocated.

Table 2. Skills needed to process tasks

SKILLS NEEDED FOR TASKS PROCESSING			
Tasks	Skills needed for task processing	Skill levels	Process time (days)
Requirements elicitation	Java	4	60
	Negotiation	5	
	Experience in telecommunications	5	
	Object oriented analysis	5	
	Tests techniques	3	
	Requirements elicitation techniques	8	
Analysis	Database	3	70
	Java	3	
	Team work	4	
	Object oriented design	4	
	Object oriented analysis	8	
	Relationship with people	5	
Design	Database	7	80
	Java	3	
	Team work	6	
	Object oriented design	7	
	Object oriented analysis	2	
Coding	Database	7	110
	Java	7	
	Team work	3	
	Experience in telecommunications	5	

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table 3. Optimized skill-based personnel allocation

SKILL-BASED PERSONNEL ALLOCATION					
Tasks	Personnel allocation	Skill levels	Process time (days) /personnel	Tasks queue time (days)	Cost (Yen)
Requirements elicitation	P2	10	5.63	1.43	140630
	P3	14	7.88	0.00	236250
	P4	15	8.44	3.66	168750
	P5	9	5.06	1.99	50625
	P6	16	9.00	2.27	135000
	total		36.00	9.35	731255
Analysis	P1	11	9.24	0.07	138600
	P2	15	12.60	0.42	315000
	P5	10	9.06	0.52	90604
	P6	14	12.69	0.83	190270
	total		43.59	1.83	734474
Design	P1	10	8.57	0.20	128570
	P4	11	9.43	3.36	188570
	P5	9	7.71	0.88	77143
	P6	10	8.57	1.97	128570
	P7	16	13.71	0.24	480000
	total		48.00	6.65	1002853
Coding	P1	11	11.52	0.04	172860
	P3	13	13.62	0.00	408570
	P4	13	13.62	0.88	272380
	P5	10	10.48	0.44	104760
	P6	16	16.76	0.62	251430
	total		66.00	1.98	1210000
Testing	P2	13	17.83	0.37	445710
	P4	13	17.83	1.23	356570
	P5	9	12.34	0.94	123430
	total		48.00	2.54	925710

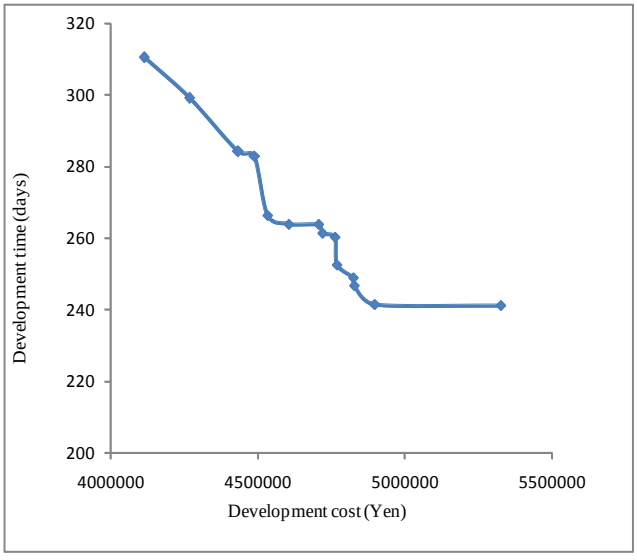

Fig.4. Pareto front : Development cost v/s time

V. CONCLUSION

Software development problem presents complex scenarios. In this study, we considered the problem of allocating the right personnel to the right job at the right time and at the right cost. We presented the skill-based allocation model in which the personnel are assigned to the tasks in accordance with the level of their skills. Any assignment that produces skill mismatch *increases* the estimated task processing time. Although team performance have an impact on the overall development result, we concentrated on the allocation of personnel so that their combined skills as a team matched the skills required to perform tasks in the software development multi-project scenario.

We simulated the schedule to determine the allocation of the limited resources to the sub-tasks and simultaneously optimized the developmental cost and the development time using the evolutionary Multi-Objective Particle Swarm Optimization (MOPSO) algorithm. The optimization result is a well-defined and well-spread Pareto front representing the trade-off between development time and development cost.

References

[1] Acuna, S.T., Juristo, N., Modeling Human Competencies in the Software Process, ProSim'03, Portland, 2003.

[2] Aickelin, U., Dowsland, K., Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem, Journal of scheduling 3 (2000) 139–153.

[3] Al-Tabtabai, H., Alex, A., Manpower scheduling optimization using genetic algorithm, in: Proceedings of the 1997 4th Congress on Computing in Civil Engineering, Philadelphia, 1997, pp. 702–709.

[4] Alvarez-Benitez, J.E., Everson, R.M., Fieldsend, J.E.: A MOPSO algorithm based exclusively on Pareto dominance concepts. In: Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization, vol. 410, pp. 459--473. Springer, Heidelberg (2005)

[5] Alvarez-Valdes, R., Crespo, E., Tamarit, J., Labour scheduling at an airport refuelling installation, Journal of the Operational Research Society 50 (3) (1999) 211–218.

[6] Azarmi,N., Abdulhameed, W., Workforce scheduling with constraint logic programming, BT Technology Journal 13 (1) (1995) 81–94.

[7] Bailey, J., Alfares, H., Lin, W., Optimization and heuristic models to integrate project task and manpower scheduling, Computers and Industrial Engineering 29 (1995) 473–476.

[8] Baker, E., Bodin, L., Finnegan,W., Ponder,R., Efficient heuristic solutions to an airline crew scheduling problem, IIE Transactions 11 (2) (1979) 79–85.

[9] Banks, J., Carson II, J.S., Discrete-Event System Simulation. Prentice-Hall, New Jersey (1984).

[10] Barreto, A., Barros, M., Werner, C., Staffing a Software Project: a Constraint Satisfaction Approach, EDSER'05, 2005, St. Louis, Missouri, pp.1-5.

[11] Caprara, A., Focacci, F., Lamma, E., Mello, P., Milano, M., Toth,P., Vigo,D., Integrating constraint logic programming and operations research techniques for the crew rostering problem, Software Practice and Experience 28 (1) (1998) 49–76.

[12] Cheng, B., Lee, J., Wu, J., A nurse rostering system using constraint programming and redundant modeling, IEEE Transactions on Information Technology in Biomedicine 1 (1) (1997) 44–54.

[13] Coelho, C., Lechunga, M.: MOPSO: A proposal for multiple objective particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation. pp. 1051--1056. IEE Press (2002).

[14] Coelho, C., Pulido, G., Salazar, M.: Handling multiobjectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation. Vol. 8, pp. 256-- 279 (2004)

[15] Deb, K., Optimization for engineering design: Algorithms and examples. Prentice Hall, Delhi (1995).

[16] Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist nondominated sorting genetic algorithm for multiobjective optimization: NSGA-II. Proceedings of Parallel Problem Solving from Nature VI Conference. pp. 849–858 (2000)

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[17] Deb, K., *Multi-objective optimization using evolutionary algorithms*, John Wiley & Sons, London, 2001.
[18] Dowsland, K., Nurse scheduling with tabu search and strategic oscillation, European Journal of Operational Research 106 (2–3) (1998) 393–407.
[19] Emden-Weinert, T., Proksch, M., Best practice simulated annealing for the airline crew scheduling problem, Journal of Heuristics 5 (1999) 419–436.
[20] Englebrecht, A.P., Computational Intelligence: An Introduction, John Wiley & Sons, London, 2002
[21] Englebrecht, A.P., Fundamentals of Computational Swarm Intelligence. John Wiley & Sons, London, 2005.
[22] Ernst, A. T., Jiang, H., Krishnamoorthy, M., and D. Sier, Staff scheduling and rostering: A review of applications, methods and models, European Journal of Operational Research, Volume 153, Issue 1, 16 February 2004, Pages 3-27.
[23] Fieldsend, J.E., Singh, S.: A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. Proc. 2002 U.K. Workshop on Computational Intelligence, Birmingham, U.K. pp. 37—44 (2002)
[24] Fishman, G. S.: Principles of Discrete Event Simulation: John Wiley & Sons, New York, (1978)
[25] Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion, and Generalization. Proc. of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA (1993)
[26] Hu, X., Eberhart, R.: Multiobjective optimization using dynamic neighborhood particle swarm optimization. Proc. Congr. Evolutionary Computation, vol. 2, pp. 1677--1681, Honolulu, (2002)
[27] Hui, X., Eberhart, R.C., Shi,Y.: Particle swarm with extended memory for multiobjective optimization. Proc. 2003 IEEE Swarm Intelligence Symp. Indianapolis, IN, pp. 193--197 (2003)
[28] Jaumard, B., Semet, F., Vovor,T., A generalized linear programming model for nurse scheduling, European Journal of Operational Research 107 (1) (1998) 1–18.
[29] Kennedy J., Eberhart, R.C.: Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942--1948 (1995)
[30] Kennedy J., Eberhart, R.C.: Swarm Intelligence, Morgan Kaufmann (2001) 18. Kennedy J., Eberhart, R.C.: Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942--1948 (1995)
[31] Kennedy J., Eberhart, R.C.: Swarm Intelligence, Morgan Kaufmann (2001)
[32] Knowles, J. Corne, D.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computing. vol. 8, pp.149 --172 (2000)
[33] Lee, M.A., Esbensen, H.: Evolutionary algorithms based multiobjective optimization techniques for intelligent systems design. Biennial Conference of the North American Fuzzy Information Processing Society, CA, pp. 360-364 (1996)
[34] Li X.: A nondominated sorting particle swarm optimizer for multiobjective optimization. In: Lecture Notes in Computer Science, vol. 2723, Proc. Genetic and Evolutionary Computation—GECCO 2003—Part I, pp. 37—48, Berlin (2003)
[35] S. Mostaghim and J. Teich, "Strategies for finding good local guides in Multi-Objective Particle Swarm Optimization (MOPSO)," *Proc. 2003 IEEE Swarm Intelligence Symp*., Indianapolis, IN, Apr. 2003, pp. 26–33.
[36] Nooriafshar, M. A heuristic approach to improving the design of nurse training schedules, European Journal of Operational Research 81 (1) (1995) 50–61.
[37] Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method in multiobjective problems. Proc. 2002 ACM Symp. Applied Computing (SAC'2002), Madrid, pp. 603--607 (2002)
[38] Ray, T., Liew, K.M.: A swarm metaphor for multiobjective design optimization. Eng. Opt., vol.34, no.2, pp.141--153 (2002)
[39] Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput., vol.2, no.3, pp. 221--248 (1994)
[40] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Fed. Inst. Technol.(ETH), Zurich, Switzerland, Nov. 1999.
[41] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.* vol. 3, pp. 257–271, Nov. 1999.
[42] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, 8(2), pp. 173–195, 2000.
[43] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Proc. EUROGEN 2001. Evolutionary Methods for *Design, Optimization and Control With Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, Sept. 2001.
[44] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm", *Proceedings of EUROGEN*, 2001.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)