2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)

Reference Line Guided Pareto Local Search for Bi-objective Traveling Salesman Problem

Chao Xia
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: heaveneleven@126.com

Xinye Cai
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: xinye@nuaa.edu.cn

Zhun Fan
School of Engineering
Shantou University
Guangdong, China
Email: zfan@stu.edu.cn

Muhammad Sulaman
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: sulman0909@gmail.com

***Abstract*—In this paper, a reference line guided Pareto local search (RLG-PLS) is proposed for combinatorial bi-objective optimization problems (CBOPs). RLG-PLS uses a set of predefined reference lines to guide the search direction and maintain the diversity of the population. Two populations are evolving in RLG-PLS, i.e., 1) the external population (EP) maintains the nondominated solutions that are closest to the reference lines; and 2) a starting population (SP) stores all the starting solutions for Pareto local search. At each generation, Pareto local search is applied to search the neighborhood of each solution in SP and these neighborhood solutions are also used to update EP and then, SP is updated with the newly added solutions from EP. When no nondominated solutions can be found (i.e., SP is empty), new reference lines are inserted to guide the Pareto local search for more new nondominated solutions. In the experimental studies, RLG-PLS is compared with MOEA/D-LS (WS, TCH, PBI), NSGA-II-LS and MOMAD on bi-objective travelling salesman problem (BOTSP). The experimental results show that RLG-PLS outperforms all the compared algorithms.**

***Index Terms*—Combinatorial bi-objective optimization, Pareto local search, travelling salesman problem;**

I. Introduction

A *multiobjective optimization problem* (MOP) can be stated as follows:

$$\begin{aligned} &\text{maximize} \quad F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{subject to} \quad\quad x \in \Omega \end{aligned} \tag{1}$$

where Ω is the *decision space*, $F : \Omega \to R^m$ consists of m real-valued objective functions. The *attainable objective set* is $\{F(x)|x \in \Omega\}$. In the case when Ω is a finite set, (1) is called a combinatorial MOP (CMOP). A CMOP with two objectives is usually called CBOP.

Let $u, v \in R^m$, u is said to *dominate* v, denoted by $u \prec v$, if and only if $u_i \leq v_i$ for every $i \in \{1, \ldots, m\}$ and $u_j < v_j$ for at least one index $j \in \{1, \ldots, m\}$[1]. A solution $x^* \in \Omega$ is *Pareto-optimal* to (1) if there exists no solution $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. $F(x^*)$ is then called a *Pareto-optimal (objective) vector*. In other words, any improvement in one objective of a Pareto optimal solution is bound to deteriorate at least another objective. The set of all the Pareto-optimal solutions is called the *Pareto set* (*PS*) and the image of (*PS*) on the objective vector space is called *Pareto front* (*PF*) [17].

As finding the exact PF of a real-world CMOP is usually NP-hard [10] by nature, over the past decades, the multiobjective heuristics (e.g. Pareto local search [15], [18], multiobjective simulated annealing [21]), multiobjective meta-heursitics (e.g., multiobjective evolutionary algorithms (MOEAs) [1]–[3], [5], [9], [22] have been widely used for approximating the PF. Among them, Pareto local search (PLS), as an extension of single objective local search [14], [15], [18], has attracted a great amount of attention. PLS explores the neighbourhood of a set of nondominated solutions for approximating PF [7].

However, PLS and its variants may suffer from long convergence time for high-quality approximation of PF [7], as it preserves all the found nondominated solutions, leading to unbearable computational cost. To address the above issue, several variants of PLS (e.g., [7], [8]), as well as its hybrid algorithms (e.g., [13], [15]) have also been proposed. For example, in [8], a dynamic discretization of the objective space is adopted to help PLS converge faster. One of the most successful PLS hybrid is the two-phase Pareto local search (2PPLS) [15]. In 2PPLS, the high-quality nondominated solutions is generated by solving a number of aggregated subproblems in the first phase and PLS is then adopted for obtaining better PF approximation in the second phase. 2PPLS generally works well, however, the direct adoption of PLS in the second phase may still face the high computational overhead.

Decomposition methods have been widely used for aiding the optimization process (e.g., [11], [12], [20], [23]). The representatives of such algorithms are MOEA/D [23] and NSGA-III [6], both of which uses predefined uniformly distributed reference lines (i.e., reference points, weight vectors or direction vectors) for maintaining the diversity of the population. The selection operation in NSGA-III can be viewed as a two-phase procedure as follows. All the nondominated solutions are selected in the first phase while the nondominated solutions closest to every reference lines are further selected

[1]In the case of maximization, the inequality signs should be reversed.

978-1-5386-3221-5/17 $31.00 © 2017 IEEE
DOI 10.1109/CSE-EUC.2017.20

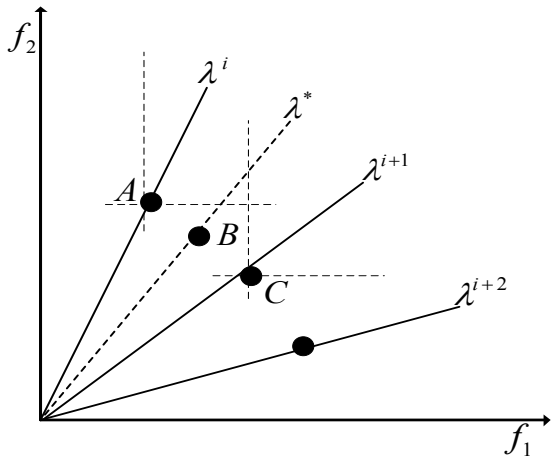


Fig. 1: An illustration of reference line guided PLS. Nondominated solution A is associated with i-th reference line and C is associated with the $(i+1)$-th reference line. A new nondominated solution B can be preserved by inserting a new reference line λ^* between λ^i and λ^{i+1}.

in the second phase for maintaining the diversity and reduce the computational overhead. In other words, the diversity is implicitly achieved by the wide spread of the reference lines and the computational complexity is also controlled by the number of reference lines.

Without knowing the shape of PF a-priori, the reference lines are usually predefined by uniformly sampling in an unit simplex [4]. The local search along a fixed set of reference line may 1) get stuck in the local optima, due to the limited neighborhood size in the local search and 2) lead to very limited number of approximated solutions, as it is very likely that multiple reference lines may lead the search to the same Pareto optimal solution.

In this paper, reference-line-guided Pareto local search (RLG-PLS) is proposed to address the above issues. The new reference lines are added to guide the Pareto local search for more nondominated solutions as well as avoiding it to be stuck in the local optima. A example of maximization problem is given in Fig. 1, where a nondominated solution B cannot be obtained due to its long distance to nondominated solution A and B, which are close to i-th and $(i+1)$-th reference lines, respectively. After a new reference line λ^* is inserted right between λ^i and λ^{i+1}, B can be preserved. Moreover, when the search along reference lines λ^i and λ^{i+1} is trapped in local optima, solution B is very likely to help them get out of local optima by searching new neighboring regions around solution B.

The rest of this paper is organized as follows. Section II describes the details of RLG-PLS. The experimental setups are given in section III. The experimental results and discussions are presented in section IV. Section V concludes this paper.

II. Algorithm

A. Main Framework

At each generation, RLG-PLS maintains two populations:

1) External population EP stores the solutions maintained by a set of reference lines W ($|W| = N$). It is worth noting that $|EP| \leq N$ as multiple reference lines may associate with the same solution.
2) Starting population SP stores the starting solutions for PLS.

To effectively insert new reference lines for guiding the local search, a flag vector $flag \in \{0,1\}^N$ is adopted to mark the effectiveness of each reference vector, where $flag_i = 1$ represents the direction along the i-th reference line once generated effective solution(s) for updating EP in the previous generations and $flag_i = 0$ represents not.

The main framework of RLG-PLS is given by **Algorithm 1**, which contains the initialization, Pareto local search and the insertion of new reference lines. Each component is explained as follows.

Algorithm 1: Main Framework

Input :
- A stopping criterion;
- N: the number of reference lines;
- A set of reference lines W;
- $N < max\_num$: the upper limit of the number of reference lines.

Output: the solution set EP.

/* Initialize all the parameters */
1 $Initialization(EP\updownarrow, SP\updownarrow, W\updownarrow)$
/* Pareto Local search on SP */
2 $ParetoLocalSearch(SP\updownarrow, EP\updownarrow, W\downarrow)$
3 If $SP == \varnothing$ and $N < max\_num$, then $InsertLines(SP\updownarrow, EP\updownarrow, W\updownarrow)$
4 If stopping criteria is satisfied, stop and output the EP. Otherwise go to Step2.

B. Initialization

In the initialization process (Algorithm 2), N reference lines are uniformly generated [4]. N solutions $\{x^1, \ldots, x^N\}$ are generated randomly or by some heuristics to initialize both EP and SP. Each reference line is associated with a solution that is closest to it. The "closeness" in this paper is defined by the acute angle between a reference line λ^j and a solution x, base on (2):

$$angle(x, \lambda^j) = arccos\left(\frac{(F(x) - z^*)^T \lambda^j}{||F(x) - z^*|| ||\lambda^j||}\right) \quad (2)$$

where $F(x) = (f_1(x), f_2(x), ..., f_m(x))^T$ is the objective vector of x, and z^* is the ideal objective vector.

Algorithm 2: Initialization

Input : EP,SP,W.
Output: EP,SP,W.

For each $k = 1, \ldots, N$, solution x^k is generated randomly or by a heuristic. x^k is associated with a reference line based on (2) .
Initialize $EP = \{x^1, \ldots, x^N\}$, $SP = EP$.
Each reference line λ^k is set as

$$\lambda^k = \left(\frac{k-1}{N-1}, \frac{N-k}{N-1}\right) \tag{3}$$

$W = \{\lambda^1, \ldots, \lambda^N\}$.
Set $flag_i = 0$, for each $i = 1, \ldots, N$.

C. Pareto Local Search

In **Algorithm 3**, Pareto local search is conducted on each solution x of SP by searching its neighborhood $N(x)$, then $N(x)$ is used to update EP (**Algorithm 4** explained in the next section). All the new nondominated solutions that successfully update EP also store in SP as the starting solutions for the next iteration of the local search. A flag variable $udpate$ is used to mark if the current solution can update EP by its generated neighborhood.

Algorithm 3: Pareto Local Search ($ParetoLocalSearch$)

Input : SP,EP,W.
Output: SP,EP.

1 $TP = EP$.
2 **for** *each* $x \in SP$ **do**
3 $update = false$;
   /* $N(x)$ is neighborhood of x generated by local search */
4 **for** *each* $x' \in N(x)$ **do**
5 $UpdateEP(x' \downarrow, EP \updownarrow, W \downarrow, update \updownarrow)$
6 **end**
7 **if** $update == true$ **then**
   /* find the index of x in EP */
8 $i = find(EP == x)$;
9 set $flag_i = 1$;
10 **end**
11 **end**
   /* SP is reinitilized by the newly added solutions of EP */
12 $SP = EP \backslash TP$.

D. Updating EP

In **Algorithm 4**, each solution y generated in the local search is used to update EP as follows. Firstly y is compared with all the other solutions in EP. y will not be stored into EP if it is dominated by any a solution in EP. Otherwise, it replaces the solution x^j associated with the reference line λ^j if it is closer to reference line λ^j(line 10-16) based on (2).

Algorithm 4: Updating EP ($UpdateEP$)

Input : y,EP,W,$update$.
Output: EP,$update$.

1 **for** *each* $x \in EP$ **do**
2 **if** $x \prec y$ **then**
3 return;
4 **end**
5 **if** $y \prec x$ **then**
6 set $x = y$;
7 $update = true$;
8 **end**
9 **end**
   /* If y is not dominated by any solution in EP */
10 **for** *each* $x^j \in EP$ **do**
11 $\theta_1 = angle(y, \lambda^j)$;
12 $\theta_2 = angle(x^j, \lambda^j)$;
13 **if** $\theta_1 < \theta_2$ **then**
14 set $x^j = y$;
15 $update = true$;
16 **end**
17 **end**

Algorithm 5: Inserting Reference Lines ($InsertLines$)

Input : SP,EP,W.
Output: SP,EP,W.

1 **for** *each* $i = 1, \ldots, N$ **do**
   /* x^i is associated with λ^i */
2 $EP = EP \cup \{x^i\}$, $W = W \cup \{\lambda^i\}$;
3 **if** $flag_i == 1$ *and* $i < N$ **then**
4 $x^* = x^i$, $\lambda^* = \lambda^i/||\lambda^i|| + \lambda^{i+1}/||\lambda^{i+1}||$;
  $EP = EP \cup \{x^*\}$, $W = W \cup \{\lambda^*\}$;
5 **end**
6 **end**
7 $SP = EP$, $N = |W|$.
8 **for** *each* $i = 1, \ldots, N$ **do**
9 set $flag_i = 0$;
10 **end**

E. Inserting New Reference Lines

When there is no starting solution for local search (i.e, $SP == \varnothing$) and the number of reference lines does not exceed the maximum value max_num (line 3 of **Algorithm 1**), new reference lines are inserted to guide the local search as follows. If any of the neighboring solution $N(x)$ that a reference line associates with are able to enter the external population EP, this reference line is called efficient (marked by $flag$). For an efficient reference line λ^i, a new reference line λ^* is inserted between i-th and $(i+1)$-th reference lines in the following way:

$$\lambda^* = \frac{\lambda^i}{||\lambda^i||} + \frac{\lambda^{i+1}}{||\lambda^{i+1}||} \tag{4}$$

III. Experimental Setups

A. Test Problems

The Traveling Salesman Problem (TSP) can be modeled as a graph G(V,E), where V =$\{1, ..., n\}$ is the set of vertices (cities) and E=$\{e_{i,j}\}_{n\times n}$ is the set of edges (connections between cities). The task is to find a Hamiltonian cycle of minimal length, which visits each vertex exactly once. In the case of the multiobjective TSP (MOTSP), each edge $e_{i,j}$ is associated with multiple values such as cost, length, traveling time, etc. Each of them corresponds to one criterion. Mathematically, the MOTSP can be formulated as:

$$\textit{minimize} \quad f_i(\pi) = \sum_{j=1}^{n-1} c^{(i)}_{\pi(j),\pi(j+1)} + c^{(i)}_{\pi(1),\pi(n)} \qquad i = 1, ..., m. \tag{5}$$

where $\pi = (\pi(1), ..., \pi(n))$ is a permutation of cities; $c^{(i)}_{s,t}$ is the cost of the edge between city *s* and city *t* regarding criterion *i*; and m is the number of objectives to be optimized. This paper also mainly focus on bi-objective Traveling Salesman Problem (BOTSP, m = 2).

In this paper, the instances of BOTSPs are used. BOTSP instances with 100 cities, published in [15], [16], [19] are adopted and named as kroAB100, euclid100, mixedAB100 and ClusterAB100.

B. Parameters Settings

Two classical and one state-of-art algorithms are adopted for comparisons as follows.

- MOEA/D-LS: MOEA/D [23] based on three decomposition methods, weighted sum (WS), Tchebycheff (TCH) and penalty boundary intersection (PBI) is adopted. For a fair comparison, MOEA/D (WS, TCH, PBI) is combined with local search heuristic (MOEA/D-LS). The number of weight vectors (reference lines) is set to $(100+400)/2 = 250$ for all instances and the neighborhood size is set to 30. For PBI, the penalty parameter θ is set to 5.
- NSGA-II-PLS: NSGA-II [6] is a classical Pareto-dominance based algorithm, which is also combined with local search heuristic for a fair comparison.
- MOMAD: Hybridization of decomposition and local search for multiobjective optimization (MOMAD) [13] is the state-of-the-art algorithm combining ideas from 2PPLS and MOEA/D.

1) The neighborhood $N(x)$ of a solution x is generated as follows. For BOTSP, a list of candidate edges (all edges of the solutions in the current EP) is maintained. More details can be referred in [13]. For a feasible solution x, two nonadjacent edges are removed and two new edges from the list are added for generating a new neighbor.
2) The number of reference lines N is initialized to 100 and its maximum allowed value max_num is set to 400 for all test instances in RLG-PLS.
3) All the compared algorithms are run independently for 20 times on each test instance.

All the compared algorithms use the same method to initialize populations. The stop criterion for all the algorithms is set as follows. An algorithm is terminated when there is no newly added solutions for local search or the number of calling $N(x)$ is up to 30000 for BOTSP instances. All algorithms are coded in C++ and the experiments are conducted on a PC equipped with Intel 2.6 GHz CPU and 8G RAM.

C. Performance Metrics

In this paper, Hypervolume indicator(I_H) [24] and Set coverage ($C-metric$) [24] are used to evaluate all algorithms.

IV. Experimental Results and Discussion

In this section, the following experiments are conducted to test the performance of RLG-PLS.

- Comparisons with MOEA/D-LS (WS, TCH, PBI), NSGA-II-PLS and MOMAD to test the performance of RLG-PLS.
- Verify the effects of inserting reference lines on the performance of RLG-PLS.

A. The comparisons of RLG-PLS with other algorithms

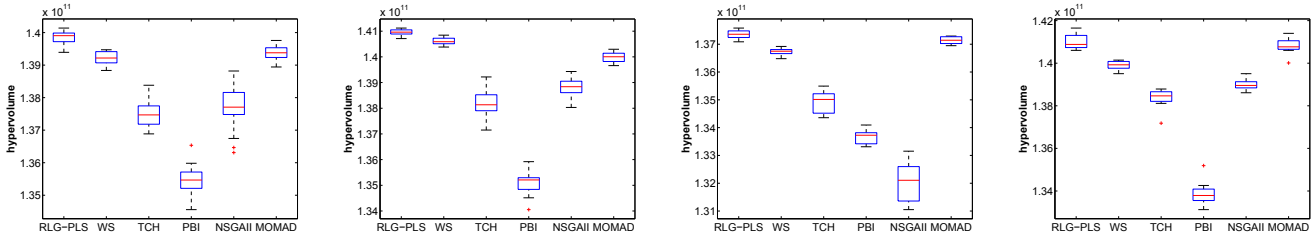
The final performance of all the compared algorithms, in terms of mean C-metric, on different BOTSP instances are presented in Table I. It can be observed very clearly that RLG-PLS outperforms all the compared algorithms on all instances. These results show that RLG-PLS has better performance on convergence.

In addition, the boxplots on the performance of all the compared algorithms in terms of hypervolume are shown in Fig. 2. RLG-PLS has the best performance on all the BOTSP instances. MOMAD also has very good performance on these instances. All the results show that RLG-PLS has better overall performance on both convergence and diversity.

The evolution of all the compared algorithms in terms of average hypervolume with the numbers of calling $N(x)$ on two BOTSP instances are plotted in Fig. 3. It can be observed that RLG-PLS converges much faster than the other compared algorithms. This can be explained as follows. In RLG-PLS, the number of reference lines (N) starts with a small number thus less solutions are used as the starting solutions for PLS and computation overhead can be reduced. Along with the increase of reference lines by inserting new reference lines, more and more nondominated solutions can be obtained. On the contrary, MOEA/D-LS (WS, TCH, PBI) maintains a fixed number of reference lines while PLS and its variants, such as MOMAD, may suffer from long convergence time for high-quality approximation of PF, as explained in Section I.

B. The Effectiveness of Inserting Reference Lines

To further verify the effectiveness of inserting reference line on the performance of RLG-PLS, three different versions of RLG-PLS are compared with each other. The first one is the original RLG-PLS, in which the number of reference lines N



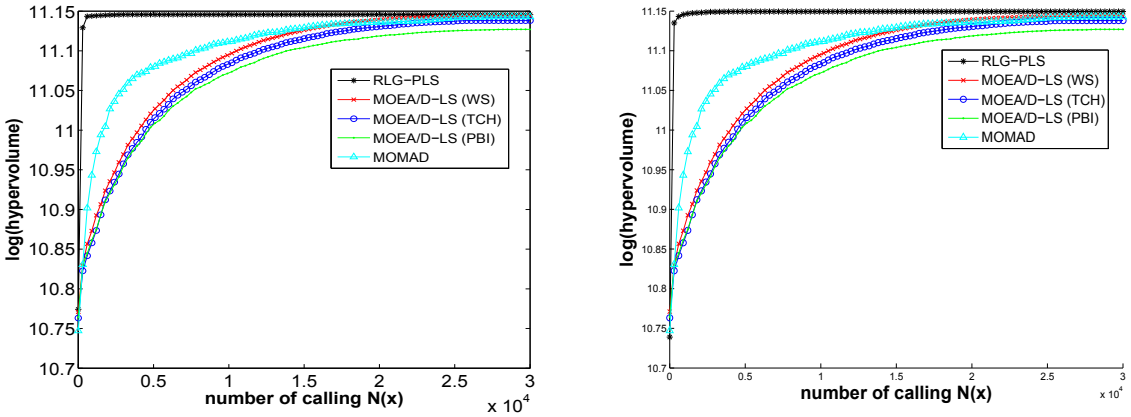
(a) kroAB100 (b) ClusterAB100 (c) euclidAB100 (d) mixedAB100

Fig. 2: Boxplots on the performance of RLG-PLS and MOEA/D-LS (WS, TCH, PBI), NSGA-II-PLS and MOMAD in terms of hypervolume on BOTSP instances.

TABLE I: The values of c-metric (%) between RLG-PLS and MOEA/D-LS (WS), MOEA/D-LS (TCH), MOEA/D-LS (PBI), NSGA-II-PLS and MOMAD on BOTSP instances

instance	MOEA/D-LS (WS)		MOEA/D-LS (TCH)		MOEA/D-LS (PBI)		NSGA-II-PLS		MOMAD	
	C(A,B)	C(B,A)	C(A,B)	C(B,A)	C(A,B)	C(B,A)	C(A,B)	C(B,A)	C(A,B)	C(B,A)
kroAB100	**92.85**	48.24	**76.67**	75.26	**86.97**	47.46	**100**	0.00	**99.96**	38.89
ClusterAB100	**79.68**	72.82	**73.20**	52.92	**84.93**	27.80	**100**	0.00	**99.86**	24.35
euclidAB100	**96.03**	48.65	**89.12**	76.60	**83.50**	46.80	**100**	0.00	**100**	0.12
mixedAB100	**99.76**	34.80	**100**	14.76	**100**	14.65	**100**	12.84	**75.55**	45.89

A corresponds to RLG-PLS.
B corresponds to the compared algorithm.



(a) kroAB100 (b) ClusterAB100

Fig. 3: Convergence graphs in terms of hypervolume (mean) obtained by MOMA/D (WS, TCH, PBI), MOMAD and RLG-PLS on two BOTSP instances.

is set to 100 and the number of maximum number of reference lines max_num is set to 400. In the second version of RLG-PLS, called RLG-PLS1, the number of reference lines N is fixed as 100. In the third version of RLG-PLS, the number of reference lines N is fixed to 600, called RLG-PLS2.

Fig. 4a show that the convergence speed of original RLG-PLS is faster than RLG-PLS1 and RLG-PLS2. And the final obtained solution set of original RLG-PLS is also better than the two others in terms of convergence and diversity performances. Therefore, it can be concluded that reference line guided technique is quite effective.

V. Conclusion

This paper proposed a reference line guided Pareto local search (RLG-PLS) for solving bi-objective combinatorial optimization problems. RLG-PLS is compared with MOEA/D-LS (WS, TCH, PBI), NSGA-II-PLS and MOMAD on bi-objective traveling salesman problems. The experimental results show that RLG-PLS outperforms the compared algorithms in term of hypervolume and set coverage metric.

Further work includes the investigations of applying RLG-PLS for solving combinatorial optimization problems with three or more objectives and more effective method of inserting reference lines.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under grant 61300159, 61175073 and 61473241, by the Natural Science Foundation of Jiangsu Province under grant BK20130808, by China Postdoctoral Science Foundation under grant 2015M571751 and by the Fundamental Research Funds for the Central Universities of China under grant NZ2013306.

References

[1] Q. Cai, M. Gong, S. Ruan, Q. Miao, and H. Du, "Network structural balance based on evolutionary multiobjective optimization: A two-step approach," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 6, pp. 903–916, 2015.

[2] C. A. Coello Coello, "Evolutionary multiobjective optimization: A historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, February 2006.

[3] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York: Springer, September 2007, iSBN 978-0-387-33254-3.

[4] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[5] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[6] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, 2014.

[7] M. M. Drugan and D. Thierens, "Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies," *Journal of Heuristics*, vol. 18, no. 5, pp. 727–766, 2012.

[8] J. Duboislacoste, M. Lopezibanez, and T. Stutzle, "Anytime pareto local search," *European Journal of Operational Research*, vol. 243, no. 2, pp. 369–385, 2015.

[9] C. M. Fonseca and P. J. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, Spring 1995.

[10] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1986.

[11] E. Hughes, "MSOPS-II: A general-purpose many-objective optimiser," in *IEEE Congress on Evolutionary Computation, CEC2007*, Sept 2007, pp. 3944–3951.

[12] E. J. Hughes, "Multiple Single Objective Pareto Sampling," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, vol. 4. Canberra, Australia: IEEE Press, December 2003, pp. 2678–2684.

[13] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization." *Cybernetics IEEE Transactions on*, vol. 44, no. 10, pp. 1808–1820, 2014.

[14] T. Lust, "Speed-up Techniques for Solving Large-scale bTSP with the Two-Phase Pareto Local Search," in *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)*. Atlanta, USA: ACM Press, July 2008, pp. 761–762, iSBN 978-1-60558-131-6.

[15] T. Lust and J. Teghem, "Two-phase pareto local search for the biobjective traveling salesman problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 475–510, 2010.

[16] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[17] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers, 1999.

[18] L. Paquete and T. Sttzle, *A Two-Phase Local Search for the Biobjective Traveling Salesman Problem*. Springer Berlin Heidelberg, 2003.

[19] G. Reinelt, "Tsplibła traveling salesman problem library," *Orsa Journal on Computing*, vol. 3, no. 1, pp. 376–384, 1991.

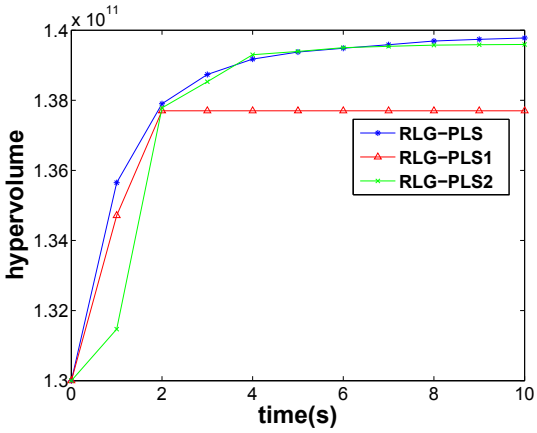
[20] J. D. Schaffer and J. J. Grefenstette, "Multiobjective Learning via Genetic Algorithms," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*. Los Angeles, California: AAAI, 1985, pp. 593–595.

[21] V. A. Shim, K. C. Tan, and H. Tang, "Adaptive memetic computing for evolutionary multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 610–621, 2015.

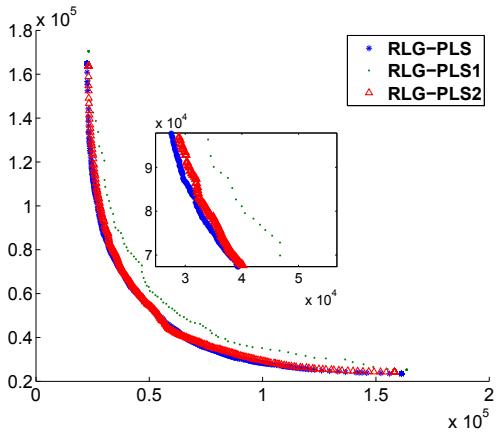
[22] K. Tan, Y. Yang, and C. Goh, "Multiobjective evolutionary algorithms and applications: Algorithms and applications." 2006.

[23] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, December 2007.

[24] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 2000.



(a) kroAB100



(b) kroAB100

Fig. 4: Convergence graphs in terms of hypervolume value and final nondominated solutions found by RLG-PLS, RLG-PLS1 and RLG-PLS2 on BOTSP instance.