Parallel Algorithm Portfolio with Market Trading-Based Time Allocation

Dimitris Souravlias, Konstantinos E. Parsopoulos and Enrique Alba

Abstract We propose a parallel portfolio of metaheuristic algorithms that adopts a market trading-based time allocation mechanism. This mechanism dynamically allocates the total available execution time of the portfolio by favoring better-performing algorithms. The proposed approach is assessed on a significant Operations Research problem, namely the single-item lot sizing problem with returns and remanufacturing. Experimental evidence suggests that our approach is highly competitive with standard metaheuristics and specialized state-of-the-art algorithms.

1 Introduction

Algorithm portfolios (APs) emerged the past two decades as a promising framework that combines different algorithms or copies of the same algorithm to efficiently tackle hard optimization problems [3, 4]. Recently, they have gained increasing attention as a general framework for incorporating different population-based algorithms to solve continuous optimization problems [7, 13]. Significant effort has been paid on the selection of the constituent algorithms of the APs [11], which may run in an independent [10] or in a cooperative way [7]. The selection is usually based on a preprocessing phase, where the constituent algorithms are selected according to their performance from a wide range of available optimization algorithms.

In parallel implementations, the minimization of the total execution time is crucial due to the limited (or expensive) resources allocated to the users in high-performance

D. Souravlias (✉) · K.E. Parsopoulos (✉)
Department of Computer Science and Engineering,
University of Ioannina, Ioannina, Greece
e-mail: dsouravl@cs.uoi.gr

K.E. Parsopoulos
e-mail: kostasp@cs.uoi.gr

E. Alba
Department of Languages and Computer Science,
University of Malaga, Malaga, Spain
e-mail: eat@lcc.uma.es

M. Lübbecke et al. (eds.), *Operations Research Proceedings 2014*,
Operations Research Proceedings, DOI 10.1007/978-3-319-28697-6_79

computer infrastructures. The AP's framework offers inherent parallelization capability that stems from the ability of concurrently using its constituent algorithms. In practice, the algorithms that are used to solve a problem in parallel are typically assigned equal execution time or function evaluation budgets that remain constant throughout the optimization process [7]. Also, it is frequently observed that different algorithms perform better in different phases of the optimization procedure or problem instances [7, 10].

Motivated by this observation, we propose an AP where the algorithms are rewarded additional execution time on a performance basis. Specifically, the portfolio adopts a trading-based mechanism that dynamically orchestrates the allocation of the total available execution time among the AP's constituent algorithms. Better-performing algorithms are assigned higher fractions of execution time compared to worse-performing ones, without modifying the AP's total execution time. The core idea behind the attained mechanism is inspired by stock trading models and involves a number of algorithms-investors that invest on elite solutions that act as stocks, using execution time as currency.

The performance of the proposed AP is evaluated on a well studied Operations Research (OR) problem, namely the single-item dynamic lot sizing problem with returns and remanufacturing [9, 12]. Its performance is compared to other metaheuristics [6] as well as state-of-the-art heuristics for the specific problem [9]. The rest of the paper is structured as follows: Sect. 2 briefly describes the problem, while Sect. 3 presents the proposed AP model. The experimental setting and results are exposed in Sect. 4 and the paper concludes in Sect. 5.

2 Problem Formulation

The considered problem constitutes an extension of the well-known Wagner-Whitin dynamic lot sizing problem [14]. It employs the dynamic lot sizing model with separate manufacturing and remanufacturing setup costs as it was introduced in [12] and further studied in [9]. The problem assumes a manufacturer that sells a single type of product over a finite planning horizon of T time periods. In each time period $t = 1, 2, \ldots, T$, the consumers state their demand denoted by D_t, along with a number of used products that are returned to the manufacturer. The fraction R_t of returned products in period t that can be recovered and sold as new is stored at a recoverables inventory with a holding cost h^R per unit time. To satisfy the demand, a number of z_t^R and z_t^M products are remanufactured and manufactured, respectively, in period t and then brought to a serviceables inventory with a holding cost h^M per unit time. Naturally, the manufacturing and remanufacturing process incur setup costs denoted by K^R and K^M, respectively.

The target is to minimize the incurring setup and holding costs by determining the exact number of manufactured and remanufactured items per period under a number of constraints. The corresponding cost function is defined as follows [9]:

$$C = \sum_{t=1}^{T} \left(K^R \gamma_t^R + K^M \gamma_t^M + h^R y_t^R + h^M y_t^M \right), \tag{1}$$

where γ_t^R and γ_t^M are binary variables denoting the initiation of a remanufacturing or manufacturing lot, respectively. The inventory levels of items that can be remanufactured or manufactured in period t are denoted by y_t^R and y_t^M, respectively. The operational constraints of the model are defined as follows:

$$y_t^R = y_{t-1}^R + R_t - z_t^R, \quad y_t^M = y_{t-1}^M + z_t^R + z_t^M - D_t, \quad t = 1, 2, \ldots, T, \tag{2}$$

$$z_t^R \le Q\, \gamma_t^R, \quad z_t^M \le Q\, \gamma_t^M, \quad t = 1, 2, \ldots, T, \tag{3}$$

$$y_0^R = y_0^M = 0, \quad \gamma_t^R, \gamma_t^M \in \{0, 1\}, \quad y_t^R, y_t^M, z_t^R, z_t^M \ge 0, \quad t = 1, 2, \ldots, T. \tag{4}$$

Equation (2) guarantees the inventory balance, while Eq. (3) assures that fixed costs are paid whenever a new lot is initiated. In [9] the value of Q is suggested to be equal to the total demand of the planning horizon. Finally, Eq. (4) asserts that inventories are initially empty and determines the domain of each variable. The decision variables of the optimization problem are z_t^M and z_t^R for each period t. Thus, for a planning horizon of T periods the corresponding problem is of dimension $n = 2T$. More details about the considered problem can be found in [6, 9, 12].

3 Proposed Algorithm Portfolio

We propose an AP that consists of metaheuristic algorithms that operate in parallel. We denote with N the number of algorithms. The AP employs a typical *master-slave* parallelization model, where each algorithm runs on a single slave node. Each algorithm invests a percentage of its assigned running time to buy solutions from the other algorithms of the AP. The remaining time is used for its own execution. We assign equal initial execution time budgets, T_{tot}, and investment time budgets, $T_{\text{inv}} = \alpha\, T_{\text{tot}}$, for all algorithms. The parameter $\alpha \in (0, 1)$ tunes each algorithm's investment policy. Clearly, high values of α indicate a risky algorithm-investor, while lower values characterize a more conservative one.

The master node retains in memory a solution archive that is asynchronously accessed by the slaves via a message passing communication mechanism. The archive holds the elite solution found by each algorithm. For their pricing, the solutions are sorted in descending order with respect to their objective values. If p_i is the position of the ith solution after sorting, then its cost is defined as $CS_i = (p_i \times SBC)/N$, where $SBC = \beta\, T_{\text{inv}}$ is a fixed base cost. The parameter $\beta \in (0, 1)$ tunes each algorithm's elitism. High values of β limit the number of the best elite solutions each algorithm can buy throughout the optimization process.

Whenever an algorithm cannot improve its elite solution for an amount of time, it requests to buy a solution from another algorithm. The master node acts as a trading broker that applies a solution selection policy to help the buyer-algorithm make the most profitable investment. In particular, the master node proposes to the buyer-algorithm elite solutions that are better than its own and cost less or equal to its current investment budget. Among the possible solutions, the algorithm opts to buy the solution that maximizes the *Return On Investment* (ROI) index, defined as $ROI_j = (C - C_j)/CS_j$, $j \in \{1, 2, \ldots, N\}$, where C is the objective value of the algorithm's own elite solution, C_j is the objective value of the candidate buying solution and CS_j is its corresponding cost. If the buyer-algorithm decides to buy the jth elite solution, it pays its price of CS_j running time to the seller-algorithm (the one that found this solution). The seller algorithm adds this time to its total execution time budget. Thus, better-performing algorithms sell solutions more often, gaining longer execution times. Yet, the total execution time of the AP remains constant.

In the present work, the proposed AP consists of 4 algorithms, namely Particle Swarm Optimization (PSO) [1], Differential Evolution (DE) [8], Tabu Search (TS) [2], and Iterated Local Search (ILS) [5].

4 Experimental Results

The proposed approach was evaluated on the established test suite used in [9]. It consists of a full factorial study of various problem instances with common planning horizon $T = 12$. Table 1 summarizes the configuration of the problem parameters

Table 1 Parameters of the considered problem and the employed algorithms

Problem parameter	Value(s)		Algorithm parameter	Value(s)
Dimension	$n = 24$	AP	Number of slave algorithms	N = 4
Setup costs	$K^M, K^R \in \{200, 500, 2000\}$		Per algorithm execution time	$T_{tot} = 75000$ ms
Holding costs	$h^M = 1, h^R \in \{0.2, 0.5, 0.8\}$		Constants α, β	$\alpha = 0.1, \beta = 0.05$
Demand for period t	$D_t \sim N(\mu_D, \sigma_D^2)$	PSO	Model	lbest (ring topology)
	$\mu_D = 100$		Swarm size	60
	$\sigma_D^2 = 10\,\%$ of μ_D (small variance)		Constriction coefficient	$\chi = 0.729$
	$\sigma_D^2 = 20\,\%$ of μ_D (large variance)		Cognitive/social constants	$c_1 = c_2 = 2.05$
Returns for period t	$R_t \sim N(\mu_R, \sigma_R^2)$	DE	Population size	60
	$\mu_R \in \{30, 50, 70\}$		Operator	DE/rand/1
	$\sigma_R^2 = 10\,\%$ of μ_R (small variance)		Differential/crossover constants	F = 0.7, CR = 0.3
	$\sigma_R^2 = 20\,\%$ of μ_R (large variance)	TS	Size of tabu list	24

Table 2 Percentage % error of the compared algorithms for different problem parameters

	Alg.	Avg	StD	Max		Alg.	Avg	StD	Max		Alg.	Avg	StD	Max
All	SM_4^+	2.2	2.9	**24.3**	$h^R = 0.2$	SM_4^+	1.7	2.5	**21.1**	$\mu_R = 30$	SM_4^+	**1.2**	**1.8**	**12.1**
	PSO	4.3	4.5	49.8		PSO	4.5	5.2	49.8		PSO	3.5	3.1	45.5
	DE	3.3	5.1	31.9		DE	3.0	5.3	30.9		DE	3.3	5.0	28.2
	TS	51.6	33.4	255.5		TS	45.0	26.4	255.5		TS	37.2	23.9	255.5
	ILS	80.3	54.3	450.8		ILS	94.8	67.2	450.8		ILS	70.6	46.5	336.8
	AP	**1.9**	**2.8**	35.6		AP	**1.5**	**2.5**	35.6		AP	1.6	2.5	25.7
$\sigma_D^2 = 10\,\%$	SM_4^+	2.1	2.8	**18.9**	$h^R = 0.5$	SM_4^+	2.3	3.0	**24.3**	$\mu_R = 50$	SM_4^+	2.3	2.7	**16.2**
	PSO	4.4	4.6	49.8		PSO	4.3	4.5	45.5		PSO	4.1	4.0	34.0
	DE	3.4	4.8	31.7		DE	3.3	5.0	31.9		DE	3.5	5.2	31.9
	TS	50.9	33.2	200.2		TS	50.8	32.1	202.1		TS	50.8	27.3	153.6
	ILS	79.7	54.2	450.8		ILS	77.6	48.2	261.1		ILS	83.7	53.0	364.2
	AP	**1.8**	**2.6**	26.8		AP	**1.9**	**2.8**	27.4		AP	**2.0**	**2.9**	27.4
$\sigma_D^2 = 20\,\%$	SM_4^+	2.4	3.0	**24.3**	$h^R = 0.8$	SM_4^+	2.8	3.0	**20.6**	$\mu_R = 70$	SM_4^+	3.3	3.5	**24.3**
	PSO	4.1	4.5	48.3		PSO	4.0	3.9	42.9		PSO	5.1	5.9	49.8
	DE	3.3	5.2	31.9		DE	3.7	4.5	31.4		DE	3.3	4.6	31.7
	TS	52.4	33.5	255.5		TS	59.1	39.0	235.2		TS	66.9	39.8	235.2
	ILS	80.9	54.4	421.5		ILS	68.4	40.8	211.4		ILS	86.5	61.1	450.8
	AP	**2.0**	**2.9**	35.6		AP	**2.2**	**3.0**	21.6		AP	**2.0**	**2.9**	35.6
$K^M = 200$	SM_4^+	**2.3**	**2.6**	**13.5**	$K^R = 200$	SM_4^+	**1.9**	**2.1**	**11.8**	$\sigma_R^2 = 10\,\%$	SM_4^+	2.2	2.9	**21.1**
	PSO	4.0	3.1	45.5		PSO	5.7	5.5	49.8		PSO	4.3	4.6	46.7
	DE	3.2	3.9	24.0		DE	3.8	4.0	24.0		DE	3.4	5.0	31.4
	TS	39.1	27.3	255.5		TS	75.2	38.0	203.3		TS	52.1	34.3	233.8
	ILS	62.6	64.0	450.8		ILS	63.0	45.4	260.4		ILS	80.4	54.4	450.8
	AP	2.4	3.0	21.6		AP	3.0	3.3	21.6		AP	**1.8**	**2.7**	35.6

(continued)

Table 2 (continued)

	Alg.	Avg	StD	Max		Alg.	Avg	StD	Max		Alg.	Avg	StD	Max
$K^M = 500$	SM_4^+	2.1	2.5	**12.8**	$K^R = 500$	SM_4^+	3.4	3.2	19.1	$\sigma_R^2 = 20\,\%$	SM_4^+	2.3	2.9	**24.3**
	PSO	4.5	4.1	27.5		PSO	3.8	4.1	37.4		PSO	4.2	4.5	49.8
	DE	2.5	2.6	15.2		DE	1.8	2.0	11.2		DE	3.3	4.9	31.9
	TS	67.9	33.1	197.1		TS	50.8	23.8	235.2		TS	51.2	32.5	255.5
	ILS	62.0	40.6	278.1		ILS	62.4	37.8	244.8		ILS	80.1	54.2	399.1
	AP	**1.8**	**2.4**	17.6		AP	**1.3**	**1.7**	**11.6**		AP	**2.0**	**2.9**	25.6
$K^M = 2000$	SM_4^+	2.3	3.4	**24.3**	$K^R = 2000$	SM_4^+	1.4	2.9	**24.3**					
	PSO	4.4	5.9	49.8		PSO	3.3	3.5	45.5					
	DE	4.3	7.1	31.9		DE	4.4	7.1	31.9					
	TS	47.9	32.7	235.2		TS	29.0	16.4	255.5					
	ILS	116.3	34.2	260.4		ILS	115.5	59.2	450.8					
	AP	**1.4**	**2.8**	35.6		AP	**1.3**	**2.8**	35.6					

as well as the employed algorithm parameters for the AP. Further details on the problem setting can be found in [9]. The proposed AP was compared against the best-performing variant (SM_4^+) of the state-of-the-art Silver-Meal heuristic [9], as well as against the sequential versions of its constituent algorithms. The goal of the experiments was to achieve the lowest possible percentage error [9] from the global optimum within a predefined budget of total execution time T_{tot}. The global optimum per problem was computed by CPLEX and was provided in the test suite.

Table 2 shows the average (Avg), standard deviation (StD), and maximum (Max) value of the percentage error for the different values of the problem parameters. A first inspection of the results reveals superiority of the proposed AP, which achieves the best overall mean percentage error (1.9 %). The second lowest value was achieved by SM_4^+ (2.2 %), followed by the sequential versions of DE (3.3 %) and PSO (4.3 %). Specifically, AP prevails in 14 out of 17 considered parameter cases, while in the rest 3 cases SM_4^+ is the dominant algorithm. The results of SM_4^+ and PSO were directly adopted from [9] and [6], respectively.

The results indicate that population-based algorithms (DE and PSO) outperform (by far) the trajectory-based ones (TS and ILS). Moreover, when all algorithms are integrated into the AP, the overall performance with respect to solution quality is further enhanced. This can be attributed to the dynamics of the trading among the algorithms. In particular, we observed that the population-based algorithms were mainly the seller ones, using their exploration capability to discover high-quality solutions. On the other hand, trajectory-based algorithms employed their exploitation power to further fine-tune the vast number of acquired solutions. From this point of view, the employed algorithms of the AP exhibited complementarity, which is a desired property in APs [7, 13]. Also, we observed that between the two population-based algorithms, PSO acquired a higher number of solutions than DE during the optimization, whereas the solutions of the latter were of better quality.

5 Conclusions

We proposed an Algorithm Portfolio (AP) of metaheuristic algorithms that operate in parallel and exchange solutions via a sophisticated trading-based time allocation mechanism. This mechanism favors better-performing algorithms with more execution time than the others. Also, it combines the exploration/exploitation dynamics of each individual constituent algorithm in an efficient way. We assessed our approach on a well studied OR problem. The experimental results were promising, indicating that the AP is highly competitive against its constituent algorithms, individually, as well as against a state-of-the-art algorithm of the considered problem.

References

1. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In Proceedings Sixth Symposium on Micro Machine and Human Science, pp. 39–43, Piscataway, NJ (1995)
2. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
3. Gomes, C.P., Selman, B.: Algorithm portfolio design: theory vs. practice. In: Proceedings Thirteenth conference on Uncertainty in artificial intelligence, pp. 190–197 (1997)
4. Huberman, B.A., Lukose, R.M., Hogg, T.: An economics approach to hard computational problems. Science **27**, 51–53 (1997)
5. Lourenco, H.R.: Job-shop scheduling: computational study of local search and large-step optimization methods. Eur. J. Oper. Res. **83**(2), 347–364 (1995)
6. Moustaki, E., Parsopoulos, K.E., Konstantaras, I., Skouri, K., Ganas, I.: A first study of particle swarm optimization on the dynamic lot sizing problem with product returns. In: Proceedings BALCOR 2013, pp. 348–356 (2013)
7. Peng, F., Tang, K., Chen, G., Yao, X.: Population-based algorithm portfolios for numerical optimization. IEEE Trans. Evol. Comput. **14**(5), 782–800 (2010)
8. Price, K.: Differential evolution: a fast and simple numerical optimizer. In: Proceedings NAFIPS'96, pp. 524–525 (1996)
9. Schulz, T.: A new silver-meal based heuristic for the single-item dynamic lot sizing problem with returns and remanufacturing. Int. J. Prod. Res. **49**(9), 2519–2533 (2011)
10. Shukla, N., Dashora, Y., Tiwari, M., Chan, F., Wong, T.: Introducing algorithm portfolios to a class of vehicle routing and scheduling problem. In: Proceedings OSCM 2007, pp. 1015–1026 (2007)
11. Tang, K., Peng, F., Chen, G., Yao, X.: Population-based algorithm portfolios with automated constituent algorithms selection. Inf. Sci. **279**, 94–104 (2014)
12. Teunter, R.H., Bayindir, Z.P., Van den Heuvel, W.: Dynamic lot sizing with product returns and remanufacturing. Int. J. Prod. Res. **44**(20), 4377–4400 (2006)
13. Vrugt, J.A., Robinson, B.A., Hyman, J.M.: Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Trans. Evol. Comput. **13**(2), 243–259 (2009)
14. Wagner, H.M., Whitin, T.M.: Dynamic version of the economic lot size model. Manag. Sci. **5**(1), 88–96 (1958)